UNIVERSITÄT LEIPZIG
Fakultät für Mathematik und Informatik
Institut für Informatik

Max-Planck-Institut für Mathematik in den Naturwissenschaften

Diplomarbeit

Reverse Engineering Methoden zur Rekonstruktion von Genregulationsnetzwerken aus Genexpressionsdaten

Antje Müller
Studiengang Medizinische Informatik

Leipzig, 5. Juli 2004

Betreuung:
Prof. Dr. rer. nat. Jürgen Jost (MPI)

Zusammenfassung

Neue Technologien auf dem Gebiet der Molekularbiologie ermöglichen es, das Expressionsverhalten mehrerer tausend Gene gleichzeitig zu untersuchen. Einen wichtigen Ansatz zur Analyse der dabei gewonnenen Genexpressionsdaten bilden Reverse Engineering Methoden. Sie versuchen, regulatorische Interaktionen zwischen den Genen aufzudecken und mit der Rekonstruktion des zugrundeliegenden genetischen Netzwerks das komplexe Zusammenspiel der Gene zu verstehen.
Das Ziel der vorliegenden Arbeit ist es, einen umfassenden Überblick über diesen Ansatz der Datenanalyse zu vermitteln. Der erste Teil der Arbeit betrachtet zunächst die theoretischen Aspekte der Reverse Engineering Methoden, um so dem Leser eine praktische Anwendung zu erleichtern und ihm beim Verständnis ausgewählter Ansätze zu helfen. So werden in diesem ersten, theoretischen Teil mögliche genetische Netzwerkmodelle vorgestellt, welche zur Beschreibung der Genexpressions- und Genregulationsprozesse dienen, und verschiedene Reverse Engineering Algorithmen detailliert beschrieben, die die Parameter eines Netzwerkmodells mit Hilfe der gegebenen Expressionsdaten bestimmen und damit festlegen, zwischen welchen Komponenten des Netzwerks regulatorische Einflüsse bestehen. Aufgrund der Begrenzung derzeit bereitstehender Expressionsdaten sowohl in Bezug auf den Datenumfang als auch bezüglich der experimentell verfügbaren Datentypen – in der Regel sind nur die mRNA-Konzentrationen gegeben, denn diese sind wesentlich einfacher und genauer zu messen als die Protein-Konzentrationen –, muß bei der Definition eines Netzwerkmodells stark von der biologischen Realität abstrahiert werden. Weitere theoretische Überlegungen betreffen die Integration von Vorwissen als eine wichtige Strategie zur Unterstützung der Rekonstruktion eines genetischen Netzwerks.
Alle vorgestellten Reverse Engineering Algorithmen wurden implementiert und können im zweiten, praktischen Teil dieser Arbeit ausführlich getestet werden. Zunächst erfolgen die Untersuchungen auf Basis von Simulationsdaten. Sie liefern wichtige Einblicke in das grundlegende Verhalten der Algorithmen in Abhängigkeit von verschiedenen Eigenschaften des zu rekonstruierenden Netzwerks und der verfügbaren Daten. Ein Anwendungsbeispiel testet schließlich alle Reverse Engineering Methoden auch an realen Expressionsdaten. Es vermittelt so abschließend einen Eindruck davon, inwieweit die auf abstrakten Netzwerkmodellen basierenden Reverse Engineering Methoden die Identifizierung von regulatorischen Einflüssen aus derzeit verfügbaren Expressionsdaten überhaupt ermöglichen und verdeutlicht die Probleme bei der praktischen Anwendung der Methoden.

Danksagung

An dieser Stelle möchte ich mich bei allen Beteiligten für das Gelingen dieser Arbeit bedanken.

Besonderer Dank gebührt Prof. Dr. Jürgen Jost für die Möglichkeit, diese Arbeit zu schreiben, für die gute Betreuung und seine stetige Bereitschaft, sich mit meinen Fragen und Problemen auseinanderzusetzen.
Prof. Dr. Friedemann Horn danke ich für die Bereitstellung der Expressionsdaten und die Beantwortung all meiner Fragen zum biologischen Hintergrund des Anwendungsbeispiels.
Ein ganz besonderer Dank geht außerdem an Kristin Missal für die unzähligen anregenden Diskussionen, die vielen hilfreichen Anmerkungen und Hinweise, ihre Geduld und ihre aufmunternden Worte.
Des weiteren bedanke ich mich bei Wenke Seifert, die stets bereitwillig und mit viel Geduld meine Fragen zum Thema Biologie, insbesondere zu den experimentellen Technologien der Molekularbiologie, beantwortete und mir geeignete Literatur zur Verfügung stellte.

Nicht zuletzt möchte ich an dieser Stelle die Gelegenheit nutzen und meinen Eltern ganz herzlich danken, die mir mein Studium ermöglicht und mich stets nach Kräften unterstützt haben!

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Reverse Engineering Methoden zur Rekonstruktion eines Genregulationsnetzwerks nutzen mathematische, statistische und informationstechnologische Modelle und Algorithmen zur Analyse von Genexpressionsdaten aus biologischen Experimenten und stellen so ein wichtiges Hilfsmittel der Molekularbiologie dar.
Eine wichtige Zielstellung der Molekularbiologie ist das Verständnis der Funktionsweise eines Organismus auf molekularer Ebene. Der Schwerpunkt liegt dabei besonders in der Aufklärung der Struktur, der Biosynthese, der Funktion und des Zusammenspiels verschiedener Makromoleküle (Desoxyribonukleinsäure, Ribonukleinsäure, Proteine). Dadurch möchte man Erkenntnisse erlangen, wie wesentliche phänotypische Eigenschaften eines Organismus durch die in bestimmten Makromolekülen (Desoxyribonukleinsäure) verschlüsselte genetische Information und die ablaufenden molekularen Prozesse festgelegt werden.
Gene bilden als wichtige Einheit der genetischen Information einen Ansatzpunkt für experimentelle Untersuchungen. Interessante Fragestellungen betreffen neben der Funktion eines Gens vor allem die differentielle Genexpression und Genregulation, denn diese bilden den Kern biologischer Anpassungs- und Entwicklungsvorgänge. Traditionelle Untersuchungsmethoden der Molekularbiologie sind zeit- und kostenintensiv. Sie arbeiten auf einer lokalen Ebene und sammeln Daten über jeweils ein bestimmtes Gen, ein bestimmtes Genprodukt oder einen bestimmten molekularen Prozeß. Gewonnene Informationen können helfen, die Funktion eines Gens festzulegen oder regulatorische Einflüsse auf die Expression eines Gens nachzuweisen.
Einen großen Fortschritt auf diesem Gebiet brachte die Entwicklung neuer Technologien, die es erlauben, mehrere tausend Gene gleichzeitig zu untersuchen und somit eine Fülle von neuen Daten zu produzieren. Man hofft, neben den herkömmlichen Fragen jetzt auch Fragen nach dem komplexen Zusammenspiel der Makromoleküle beantworten zu können. Die entsprechende Auswertung dieser Daten erfordert den Einsatz verschiedener Analyse- und Modellierungsmethoden.
Einen wichtigen Ansatz bilden hier Reverse Engineering Methoden. Sie verfolgen das Ziel, regulatorische Interaktionen zwischen den Makromolekülen aufzudecken und mit der Rekonstruktion des zugrundeliegenden genetischen Netzwerks das komplizierte Zusammenspiel dieser molekularen Strukturen zu verstehen. Die Komplexität biologischer Regulationsmechanismen, aber auch die Begrenzung experimenteller Daten erfordern dabei eine abstrakte Modellierung der ablaufenden Prozesse.

Die vorliegende Arbeit soll einen umfassenden Überblick über diesen Ansatz der Datenanalyse vermitteln. Sie untergliedert sich in sieben Kapitel:
Das 1. Kapitel bietet eine Einführung in das Thema. Es werden wichtige biologische Grundlagen beschrieben, experimentelle Verfahren vorgestellt und der Begriff des Reverse Engineerings eingeführt.
Die theoretischen Aspekte der Reverse Engineering Methoden behandeln Kapitel 2 und 3. Sie sollen dem Leser bei dem Verständnis ausgewählter Reverse Engineering Ansätze helfen und eine praktische Umsetzung erleichtern. Zunächst befaßt sich das 2. Kapitel mit der Wahl eines geeigneten Netzwerkmodells zur Modellierung eines Genregulationsnetzwerks. Wichtige Eigenschaften eines genetischen Netzwerkmodells werden besprochen und ausgewählte Modelle vorgestellt. Derzeit verfügbare Daten stammen meist aus Microarray-Experimenten zur Quantifizierung der mRNA-Konzentrationen. Die Arbeit beschränkt sich hier deshalb auf abstrakte Netzwerkmodelle, die nur die mRNA-Konzentrationen der Gene modellieren. Netzwerkmodelle, die zusätzlich die Protein-Konzentrationen integrieren, werden dagegen nicht betrachtet. Anschließend behandelt das 3. Kapitel den Einsatz von Reverse Engineering Algorithmen, die die Parameter eines Netzwerkmodells mit Hilfe der gegebenen Expressionsdaten festlegen und spezifizieren, zwischen welchen Netzwerkkomponenten regulatorische Einflüsse bestehen. Für jedes der vorgestellten Netzwerkmodelle erfolgt zunächst eine kurze Analyse der speziellen Aufgaben eines Reverse Engineering Algorithmus. Ein jeweils ausgewählter Algorithmus soll dann detailliert beschrieben, auf seine Implementierung eingegangen sowie seine Limitationen untersucht werden.
Eine wichtige Strategie zur Rekonstruktion von Genregulationsnetzwerken stellt die Integration von Vorwissen über die Struktur des zu rekonstruierenden Netzwerks dar, denn eine Kombination dieses Wissens mit den Informationen aus gegebenen Expressionsdaten kann den Reverse Engineering Prozeß maßgeblich unterstützen. Kapitel 4 beschäftigt sich deshalb mit möglichen Ansätzen zur Integration von Vorwissen in den Reverse Engineering Prozeß.
Alle in Kapitel 3 betrachteten Reverse Engineering Algorithmen wurden implementiert und zunächst an Simulationsdaten getestet. Die daraus resultierenden Ergebnisse werden im 5. Kapitel ausführlich diskutiert und vergleichend nebeneinander gestellt. Sie liefern einen Einblick in das Verhalten der Algorithmen in Abhängigkeit von verschiedenen Eigenschaften des zu rekonstruierenden Netzwerks und der verfügbaren Expressionsdaten.
Schließlich testet Kapitel 6 die Reverse Engineering Methoden in einem Anwendungsbeispiel auch an realen Expressionsdaten und vermittelt dem Leser damit einen Eindruck, inwieweit die auf abstrakten Netzwerkmodellen basierenden Reverse Engineering Methoden die Rekonstruktion eines Genregulationsnetzwerks überhaupt ermöglichen.
Das 7. Kapitel faßt abschließend die Ergebnisse dieser Diplomarbeit zusammen und gibt einen kurzen Ausblick auf zukünftige Weiterentwicklungen.

Kapitel 1

Biologische und technische Grundlagen

Für das Verständnis der in dieser Arbeit behandelten Problematik ist es hilfreich, sich mit dem zugehörigen biologischen Hintergrund auseinanderzusetzen. Der erste Abschnitt dieses Kapitels dient deshalb zur Definition grundlegender Begriffe und verschafft so einen kurzen Einblick in die Grundlagen der Genexpression und Genregulation.
Im Anschluß daran wird ein kurzer Überblick über die experimentellen Technologien zur Generierung von Expressionsdaten gegeben.
Auf Basis der so gelegten Grundlagen erfolgt im dritten Abschnitt eine Einführung des Reverse Engineering Begriffs.

1.1 Biologische Grundlagen

Die Gesamtheit aller Gene eines Organismus bezeichnet man auch als dessen Erbsubstanz. In ihr ist die vererbte Information des Organismus verschlüsselt. Träger der Erbinformation sind die, bei Eukaryonten im Zellkern einer jeden Zelle, bei Prokaryonten im Zellplasma befindlichen Chromosomen, deren wichtigster Bestandteil die Desoxyribonukleinsäure (*engl.: desoxyribonuclein acid - DNA*) ist. Dieses Polymer ist aus vier verschiedenen Nukleotiden zusammengesetzt. Ein Nukleotid besteht dabei aus einem Zucker – der Desoxyribose –, aus einem Phosphorsäurerest und aus einer der vier Stickstoffbasen Adenin (A), Guanin (G), Cytosin (C) und Thymin (T). Ein DNA-Strang, in dem sich diese Nukleotidbausteine in einer organismusspezifischen Abfolge milliardenfach wiederholen, wird durch einen zweiten, komplementären Strang zu einem Doppelstrang ergänzt, der sich zu einer Spirale – der sogenannten Doppelhelix – windet (Abbildung 1.1). Der Zusammenhalt entsteht dabei durch das Ausbilden von Wasserstoffbrückenbindungen zwischen den komplementären Basen Cytosin - Guanin und Adenin - Thymin.

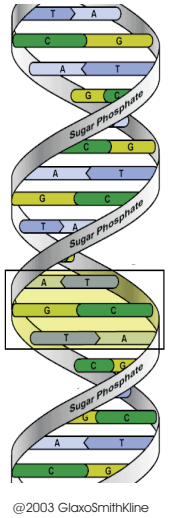


Abbildung 1.1: **DNA-Doppelhelix** [21]

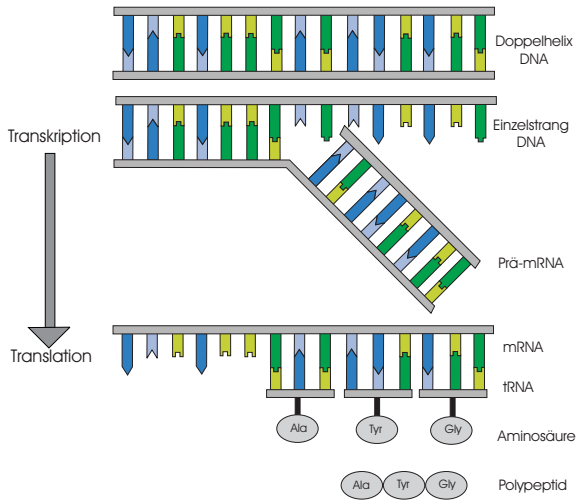


Abbildung 1.2: **Proteinsynthese**

Als Träger der Erbinformation legt die DNA die Ausprägung bestimmter Merkmale eines Organismus fest, indem sie der Zelle die Synthese der jeweiligen Proteine vorschreibt. Proteine stellen damit das Bindeglied zwischen Genotyp und Phänotyp[1] eines Organismus dar. Bei der Definition eines Gens greift man häufig auf die vereinfachte „Ein-Gen-ein-Protein“-Hypothese zurück:

Definition 1.1 (Gen) *Ein Gen ist definiert als ein Abschnitt der DNA, der die Information für die Synthese eines bestimmten Proteins kodiert.*

Der Prozeß der Proteinsynthese dient zum Auslesen und zur Dekodierung der in der DNA gespeicherten Information zur Produktion eines Proteins. Im wesentlichen wird dabei in zwei aufeinanderfolgenden Schritten die Basensequenz der DNA in die Aminosäuresequenz eines Polypeptids übersetzt (Abbildung 1.2).
Im ersten Schritt – der Transkription – schreibt ein Enzym, die RNA-Polymerase, die Basensequenz eines DNA-Abschnitts in die Basensequenz eines Messenger-Ribonukleinsäure-Moleküls (*engl.: messenger ribonuclein acid - mRNA*) um. Die mRNA ähnelt in ihrem Aufbau der DNA. Die Nukleotidbausteine bestehen hier aber aus einem anderem Zucker – der Ribose – und anstelle der Stickstoffbase Thymin (T) wird mit der Base Uracil (U) gearbeitet. Auch liegt die mRNA im Gegensatz zur DNA als Einzel-Strang vor. Das aus der Transkription hervorgegangene „Prä-mRNA“-Transkript enthält neben kodierenden Segmenten (Exons) auch nichtkodierende Segmente (Introns). Nachdem alle nichtkodierenden Segmente aus dem

[1] Als Genotyp eines Organismus bezeichnet man seine Erbsubstanz, als Phänotyp die Gesamtheit aller seiner Eigenschaften und Merkmale.

mRNA-Molekül entfernt wurden (RNA-Spleißen), wird es aus dem Zellkern hinaus in das Zellplasma zu den Ribosomen transportiert. Dort findet der zweite Schritt der Proteinsynthese – die Translation – statt, der die Basensequenz des mRNA-Moleküls in die Aminosäuresequenz eines Polypeptids übersetzt. Jeweils drei aufeinanderfolgende Basen bilden dabei ein Codon und kodieren eine Aminosäure. Die Übersetzung eines Codons erfolgt durch das Anlagern eines komplementären tRNA-Moleküls (*engl.: transfer ribonuclein acid*) – dem Anticodon –, das die entsprechende Aminosäure gebunden hat. Durch die Ausbildung von Peptidbindungen werden die Aminosäuren dann zu einer Polypeptidkette verknüpft. Im Anschluß erfolgt durch die räumliche Faltung des Polypeptids und posttranslationale Modifikationen – zum Beispiel das Anheften von Zuckern, Lipiden und Phosphatgruppen – die Bildung eines funktionstüchtigen Proteins.
Ein Gen wird exprimiert, falls die Zelle die in dem Gen kodierte Information benutzt, um mit Hilfe der einzelnen Prozesse der Proteinsynthese ein spezifisches Genprodukt zu produzieren:

Definition 1.2 (Genexpression [51]) *Unter der Genexpression versteht man die Umsetzung der in einem Gen verschlüsselten Information zu einem Genprodukt (Protein, mRNA). Sie umfaßt die Prozesse der Transkription, der Translation und alle darüber hinausgehenden, posttranslationalen Modifikationen.*

Die Charakterisierung der Expressionsrate eines Gens erfolgt durch eine Messung der Konzentrationen entstandener Genprodukte. Ein Gen kann zu verschiedenen Zeitpunkten in Abhängigkeit von bestimmten Gegebenheiten unterschiedlich stark exprimiert werden:

Definition 1.3 (differentielle Genexpression [35]) *Unter differentieller Genexpression versteht man die Expression eines Gens in Abhängigkeit von zeitlichen, räumlichen, zelltypspezifischen, organspezifischen und signalvermittelten Parametern.*

Andere Gegebenheiten bewirken also die Aktivierung sonst stillgelegter Gene oder die Repression aktiver Gene. Dadurch ist die Zelle in der Lage, sich verschiedenen äußeren und physiologischen Reizen anzupassen. Gleichzeitig befähigt die differentielle Genexpression mehrzellige Eukaryonten, aus einer Stammzelle unzählige Zelltypen mit verschiedenen spezialisierten Funktionen zu entwickeln. Von entscheidender Bedeutung für die differentielle Genexpression ist ihre Regulation:

Definition 1.4 (Genregulation [51]) *Genregulation ist die Veränderung der Art und/oder der Geschwindigkeit von zellulären Prozessen durch eine Kontrolle der Aktivität bestimmter Gene, um einzelne biochemische Reaktionen an gegebene Situationen anzupassen.*

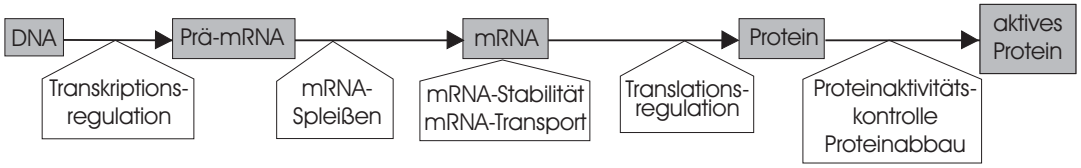


Abbildung 1.3: **Ebenen der Genregulation:** Schritt 1 - Regulation bei der Transkription; Schritt 2 - Regulation durch alternatives Spleißen; Schritt 3 - Regulation auf der Ebene der mRNA durch Stabilisations- und Zerfallsprozesse sowie durch Beschränkung des mRNA-Transports; Schritt 4 - Regulation bei der Translation; Schritt 5 - Regulation durch Protein-Abbau, Proteinaktivierung und -deaktivierung

Genregulation findet in den verschiedenen Ebenen der Genexpression statt: während der Transkription, beim RNA-Spleißen und RNA-Transport, während der Translation und auch in den posttranslationalen Modifikationen (Abbildung 1.3). Häufig erfolgt sie auf der Ebene der Transkription (Abbildung 1.3 - Schritt 1). Bei Eukaryonten dominiert dabei die positive Transkriptionsregulation [35]: RNA-Polymerasen binden an Bereiche der DNA, die als Promotor bezeichnet werden, um von dort die Transkription eines Gens zu initiieren. Allerdings können sie allein weder einen Promotor erkennen, noch an ihn binden. Sogenannte Transkriptionsfaktoren müssen zuerst an den Promotor oder anderen Kontrollregionen der DNA (Enhancer) andocken, bevor die RNA-Polymerase an den Promotor binden kann. Transkriptionsfaktoren sind dabei Proteine in einer aktiven Form – also nichts anderes als die Produkte exprimierter Gene. Das Gegenstück zu dieser positiven Transkriptionskontrolle bildet die negative Transkriptionsregulation. Transkriptionsfaktoren können durch das Anlagern an anderen Kontrollregionen der DNA (Silencer) die Transkription eines Gens auch unterdrücken. Ähnliche Regulationsmechanismen existieren bei den Prokaryonten. Über Transkriptionsfaktoren können sowohl in Eukaryonten als auch in Prokaryonten extrazelluläre Reize, z.B. Licht, Temperatur sowie Wachstums-, Überlebens- und Todesfaktoren durch Signaltransduktion Einfluß auf die Transkription eines Gens nehmen [35].
Manche aus der Transkription hervorgegangenen „Prä-mRNA“-Moleküle können auf mehrere Arten gespleißt werden (Alternatives Spleißen), so daß verschiedene mRNA-Moleküle als Spleißprodukte entstehen, die wiederum in unterschiedliche Proteine umgesetzt werden (Abbildung 1.3 - Schritt 2). Der Mechanismus des Alternativen Spleißens steht damit im Widerspruch zur vereinfachten „Ein-Gen-ein-Protein“- Hypothese.
Weitere Regulationsmechanismen ergeben sich auf der Ebene der mRNA (Abbildung 1.3 - Schritt 3). Stabilisations- und Zerfallsprozesse regulieren die Konzentration des mRNA-Transkripts eines Gens. Nur ein Teil der produzierten mRNA gelangt aus dem Zellkern in das Zellplasma.
Weiterhin wird die Genexpression während der Translation des mRNA-Transkripts

in ein Polypeptid reguliert. (Abbildung 1.3 - Schritt 4). Hier erfolgt die Kontrolle ähnlich wie bei der Transkription bei der Initiierung des Translationsprozesses.
Posttranslationale Modifikationen legen maßgeblich die funktionalen Eigenschaften eines Proteins fest. Sie können so regulieren, ob ein Protein in einer inaktiven Form vorliegt oder in einer aktiven Form, in der es spezifische regulatorische Aufgaben erfüllen kann. Die Aktivierung spezifischer Proteinfaktoren (Abbildung 1.3 - Schritt 5) wird dabei meist durch die Mechanismen der Signaltransduktion ausgelöst. So beruht die Signalwirkung der meisten Hormone, Zytokine und anderer Substanzen auf deren Bindung an bestimmte Rezeptoren und der folgenden Phosphorilierung von Proteinen. Diese werden dann entweder selbst als Transkriptionsfaktor aktiv oder bewirken über zum Teil sehr komplexe Signalketten die Aktivierung anderer Proteinfaktoren [35].
Schließlich kann auch durch die Stabilität der aus der Translation hervorgegangenen Proteine eine Kontrolle der Genexpression ausgeübt werden (Abbildung 1.3 - Schritt 5).
Alles in allem entsteht so ein komplexes Netzwerk aus regulatorischen Interaktionen zwischen DNA, RNA, Proteinen und anderen Substanzen:

Definition 1.5 (Genregulationsnetzwerk) *Unter einem Genregulationsnetzwerk versteht man das komplexe Zusammenspiel der in einer Zelle vorkommenden Derivate der einzelnen Gene (DNA, mRNA, inaktives Protein, aktives Protein, etc.), sowie anderen intra- und extrazellulären Einflußfaktoren (Licht, Temperatur, Wachstums-, Überlebens- und Todesfaktoren, Stoffwechselprodukte, Nährstoffe, etc.), die durch regulatorische Einflüsse aufeinander einwirken.*

1.2 Experimentelle Bestimmung des Expressionsverhaltens

Wichtige Größen zur Charakterisierung des Expressionsverhaltens der Gene einer Zelle sind vor allem die entsprechenden mRNA- und Protein-Konzentrationen, denn die Expressionsrate eines Gens kann maßgeblich durch diese Größen beschrieben werden.
Herkömmliche Methoden der Molekularbiologie zur Gewinnung solcher Expressionsdaten arbeiten nach dem zeit- und kostenintensiven „Ein-Gen-ein-Experiment“-Prinzip. Sie erlauben lediglich die Analyse eines einzigen Gens und erschweren damit ein umfassendes Verständnis des komplexen Zusammenspiels der einzelnen Komponenten des Genregulationsnetzwerks einer Zelle.
Einen erheblichen Vorteil brachte die Entwicklung sogenannter „Large-Scale“-Experimente, die es ermöglichen, die mRNA- und Protein-Konzentrationen mehrerer tausend Gene gleichzeitig zu bestimmen und damit die expressionelle Aktivität einer Zelle zu einem beliebigen Zeitpunkt in einem bestimmten physiologischen Zustand

zu charakterisieren. In diesem Unterabschnitt sollen kurz die derzeit verfügbaren Technologien zur Quantifizierung der mRNA- und Proteinkonzentrationen vorgestellt werden.

1.2.1 Quantifizierung der mRNA-Konzentrationen

Bekannte Verfahren zur Bestimmung der mRNA-Konzentrationen in einer Zelle sind neben der Microarray-Technologie auch die SAGE Methode und eine Kombination aus RT-PCR und Northern-Hybridisierung.

Microarray-Technologie

Die fundamentale Basis der Microarray-Technologie [59] bildet die sogenannte Hybridisierung: Zwei DNA-Stränge bzw. ein DNA- und ein RNA-Strang hybridisieren miteinander, wenn sie komplementär zueinander sind. Dazu bilden sich zwischen den komplementären Basen Guanin-Cytosin und Adenin-Thymin bzw. Adenin-Uracil Wasserstoffbrückenbindungen aus, die die Stränge miteinander verknüpfen.
Microarray-Chips sind kleine Glasträger, auf denen tausende verschiedene Nukleotidsequenzen (Proben) mit bekannter Basenfolge in einer hohen Dichte befestigt wurden. Dabei ist in jeder Probe die entsprechende Nukleotidsequenz in genügend hoher Anzahl vorhanden, um ausreichend Bindungsmöglichkeiten zur Verfügung zu stellen. Eine spätere Zuordnung ist möglich, da die Position einer jeden Probe genau bekannt ist.
Die in einer einzigen Zelle enthaltenen Mengen verschiedener mRNA-Moleküle sind viel zu klein, um sie mit diesem Verfahren zu quantifizieren. Man arbeitet deshalb in einem Microarray-Experiment mit einer ganzen Zellpopulation gleichartiger Zellen. Für die Durchführung eines solchen Experiments ist es zunächst erforderlich, die mRNA-Moleküle aus den Zellen der zu untersuchenden Population zu extrahieren und mit einem Fluoreszenzfarbstoff zu markieren. Anschließend werden sie dann in einer wäßrigen Lösung auf den Microarray-Chip aufgebracht und können mit den entsprechenden komplementären Probe-Sequenzen hybridisieren. Nach der Entfernung aller nicht gebundenen mRNA-Moleküle kann man mit Hilfe eines Laserscanners durch die Stärke der Fluoreszenz einer bestimmten Probe analysieren, wieviele komplementäre mRNA-Moleküle eines bestimmten Typs an die Nukleotidsequenzen der Probe gebunden haben und Rückschlüsse auf die durchschnittliche Konzentration dieses mRNA-Moleküls in den Zellen der untersuchten Population ziehen. Da die Basenfolge der Sequenzen in einer Probe bekannt ist, können die gebundenen mRNA-Moleküle einem bestimmten Gen zugeordnet werden.
Im wesentlichen unterscheidet man Oligonukleotid-Arrays und cDNA-Microarrays. Erstere enthalten als Probe-Sequenzen kurze, in der Regel 25 Basen lange Nukleotidsequenzen – sogenannte Oligonukleotide. Diese werden direkt auf dem Chip durch das schrittweise Anlagern von Nukleotiden synthetisiert. Im Gegensatz dazu ver-

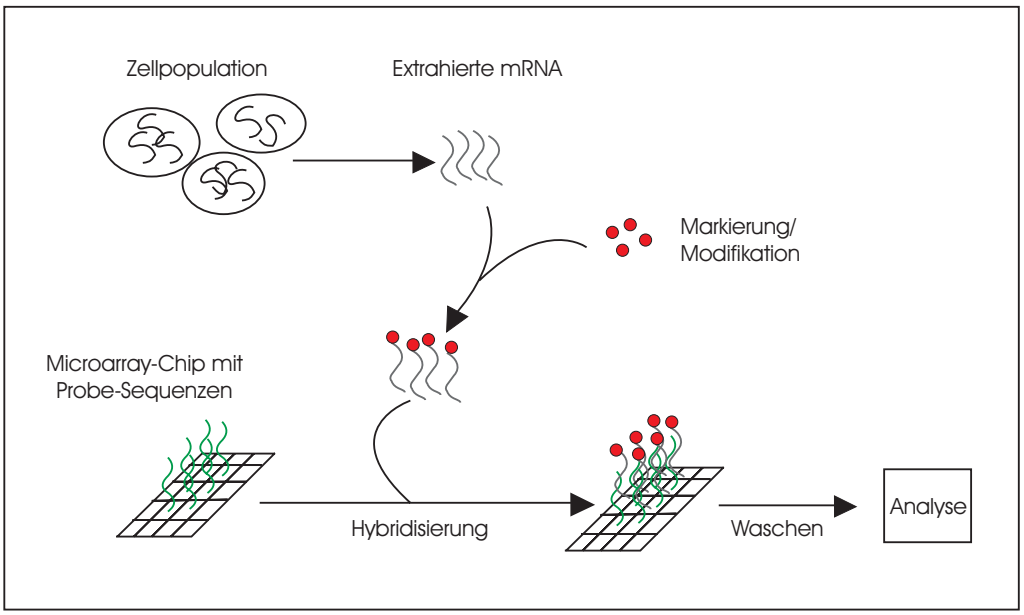


Abbildung 1.4: **Microarray-Experiment zur Quantifizierung der mRNA-Konzentrationen.**

wendet man bei den cDNA-Microarrays als Probe-Sequenzen vollständige cDNA-Sequenzen[2]. Diese werden vorher synthetisiert und mit Hilfe spezieller Technik zielgenau auf den Glasträger aufgebracht.

SAGE

Das SAGE Verfahren (Serial Analysis of Gene Expression) [14, 62] nutzt die Tatsache, daß jedes mRNA-Molekül bereits durch ein kurzes Teilstück seiner Nukleotidsequenz (zwischen 10-17 Basen lang) eindeutig identifiziert werden kann. Um also ein mRNA-Molekül über seine Basenfolge zu identifizieren, muß nicht das gesamte Molekül, sondern lediglich ein kurzes Teilstück sequenziert[3] werden.
Analog zu der Microarray-Technologie arbeitet man auch hier mit einer Zellpopulation, aus der zunächst die mRNA-Moleküle extrahiert werden müssen. Ein spezielles Enzym – die Reverse Transkriptase – schreibt dann jedes einzelne mRNA-Molekül in eine cDNA-Sequenz um. Aus diesen cDNA-Molekülen werden anschließend kurze Teilstücke an einer spezifischen Position entfernt. Diese dienen zur Identifizierung der mRNA-Moleküle. Damit nicht jedes dieser Teilstücke einzeln sequenziert werden

[2]Eine cDNA-Sequenz (complementary DNA) ist ein DNA-Molekül, das aus einem mRNA-Molekül generiert wurde. Es ist komplementär zur Sequenz dieses mRNA-Moleküls und besteht nur aus kodierenden Abschnitten.

[3]Sequenzieren bedeutet, mit Hilfe spezieller Technik – sogenannter Sequenziermaschinen – die Nukleotidsequenz eines Moleküls auszulesen.

muß, verbindet man sie miteinander vor der Sequenzierung zu großen Molekülen, den sogenannten Concatemeren. Die Anzahl, mit der ein bestimmtes Teilstück in diesen Concatemeren vorkommt, erlaubt dann Rückschlüsse auf die Konzentration des zugehörigen mRNA-Moleküls; die Basenfolge des Teilstücks identifiziert das mRNA-Molekül und ordnet es einem bestimmten Gen zu.

RT-PCR und Northern-Hybridisierung

Im Unterschied zu den beiden vorangegangenen Verfahren erlaubt es die Kombination dieser beiden Methoden, auch sehr kleine Mengen von mRNA-Molekülen zu analysieren. Deshalb ist es hier möglich, die Konzentrationen der verschiedenen mRNA-Moleküle einer einzigen Zelle zu bestimmen – es muß also nicht auf die Durchschnittswerte aus einer Zellpopulation zurückgegriffen werden, die zum Teil recht ungenau sind und viele Zusammenhänge zwischen dem Expressionsverhalten der einzelnen Gene verwischen.
Mittels RT-PCR (Reverse Transcriptase Polymerase Chain Reaction) [45] werden zunächst die aus einer Zelle extrahierten mRNA-Moleküle durch das Enzym Reverse Transkriptase in cDNA-Sequenzen umgeschrieben, anschließend diese cDNA-Sequenzen millionenfach vervielfältigt und so quantifizierbare Mengen produziert. Das dabei entstandene komplexe Gemisch kann dann durch Elektrophorese in die verschiedenen cDNA-Sequenzen aufgetrennt werden: Man bringt das Gemisch dazu auf einen geeigneten Träger (Gel, spezielles Papier) auf. Ein angelegtes elektrisches Feld löst die Wanderung der einzelnen cDNA-Sequenzen auf dem Träger aus. In Abhängigkeit ihrer Größe und Basensequenz wandern die verschiedenen Sequenzen mit unterschiedlichen Geschwindigkeiten und trennen sich in einzelne Banden auf. Das entstehende Elektropherogramm wird anschließend mittels der Northern-Blotting-Technik [45] auf eine Membran transferiert. Durch Zugabe radioaktiv markierter Sonden, die durch Hybridisierung an die cDNA-Sequenzen auf der Membran binden, können die Banden dann analysiert werden. Der Ort einer Bande dient zur Identifizierung der Sequenz und damit der Zuordnung zu einem bestimmten Gen. Die Intensität der Radioaktivität einer Bande beschreibt die Menge der cDNA-Sequenzen an dieser Stelle und ist proportional zur Konzentration des entsprechenden mRNA-Moleküls.
Im Gegensatz zu den beiden vorangegangenen Technologien ist dieses Verfahren allerdings weniger parallel – die Anzahl der Gene, deren zugehörige mRNA-Konzentrationen in einem Experiment bestimmt werden können, ist wesentlich geringer. Dieser Nachteil wird jedoch durch eine hohe Genauigkeit der Ergebnisse aufgrund der Arbeit mit einer einzelnen Zelle anstelle einer Zellpopulation wieder ausgeglichen.

1.2.2 Quantifizierung der Protein-Konzentrationen

Im Unterschied zu den mRNA-Molekülen ist die Identifizierung und Quantifizierung der Proteine in einer Zelle wesentlich komplizierter. Ein traditionelles Verfahren stellt hier die 2D-Elektrophorese [35] dar. Neue Möglichkeiten ergeben sich derzeit aus der Entwicklung sogenannter Protein-Arrays [40].

2D-Elektrophorese

Wie bereits für cDNA-Sequenzen beschrieben wird auch das aus einer Zelle extrahierte Proteingemisch in einem Gel durch das Anlegen eines elektrischen Feldes in seine unterschiedlichen Proteine aufgetrennt. Die einzelnen Proteine bilden in Abhängigkeit ihrer Größe und ihres molekularen Gewichts ein sichtbares Bandenmuster aus. Auch hier kann dann die Auswertung dieses Elektropherogramms mit Hilfe einer Blotting-Technik – des Western-Blottings [45] – erfolgen. Die Proteine werden dazu auf elektrophoretischem Wege in eine proteinbindende Membran transferiert, wo man sie mittels spezifischer, mit einem Farbstoff markierter Antikörper nachweisen kann. Der Ort einer Bande dient wieder zur Identifizierung des dort angesammelten Proteins; die Intensität der Bandenfärbung gibt Auskunft über die Konzentration des jeweiligen Proteins.
Ein Nachteil dieses Verfahrens ergibt sich aus der Tatsache, daß für viele Banden noch nicht bekannt ist, welches Protein sie repräsentieren. Damit können nicht alle Proteine des extrahierten Proteingemischs identifiziert werden. Weiterhin ist oftmals auch die Auflösung nicht groß genug, um alle Proteine zu separieren und eine Bande des Elektropherogramms repräsentiert dann mehr als ein Protein.

Protein-Arrays

Die Entwicklung von Protein-Arrays sollte die Vorteile der Microarray-Technologie auf die Proteinanalyse übertragen. Es werden hier komplette Proteine als Zielmoleküle auf die Oberfläche eines Chips aufgebracht. Über Protein-Protein-Interaktionen reagieren die aus einer Zellpopulation extrahierten Proteine dann mit diesen Zielmolekülen und können so identifiziert und quantifiziert werden. Die Prinzipien der Herstellung und Bindung gestalten sich hier allerdings viel komplizierter und es ergeben sich eine Reihe von Problemen.
Zum einen sind Proteine im Gegensatz zu den robusten Nukleotidsequenzen stark abhängig von den äußeren Bedingungen (Temperatur, ph-Wert, Ionenstärke, etc.). Die Änderung dieser Bedingungen hat häufig die Denaturierung und den Verlust der Aktivität der Proteine zur Folge. Zum anderen ist die Funktion eines Proteins abhängig von seiner spezifischen, sehr empfindlichen 3D-Struktur. Damit die Proteine an die Zielmoleküle auf dem Chip binden können, muß die dreidimensionale Struktur dieser Moleküle erhalten bleiben. Weiterhin sind die Interaktionen zwischen den Proteinen sehr heterogen und haben unterschiedliche Eigenschaften. Im

Gegensatz zu den Nukleotidsequenzen sind zum Beispiel die entsprechenden Bindungsstärken und -stabilitäten nicht standardisiert.
Die Herstellung solcher Protein-Arrays gestaltet sich daher sehr schwierig. Als Folge ist ihr Einsatz momentan noch stark beschränkt – zur Quantifizierung von Proteinkonzentrationen wird meist auf das traditionelle Verfahren der Elektrophorese zurückgegriffen.

1.3 Reverse Engineering

Unter dem Begriff Reverse Engineering verbirgt sich hier die Aufgabenstellung, aus den gegebenen Expressionsdaten, die das experimentell bestimmte Expressionsverhalten der Gene beschreiben, Rückschlüsse auf die regulatorischen Interaktionen zwischen den Komponenten des Genregulationsnetzwerks einer Zelle (siehe Definition 1.5) zu ziehen. Es sollen also Informationen darüber gewonnen werden, wie die Expression eines Gens durch die Expression anderer Gene, aber auch durch extra- und intrazelluläre Einflußfaktoren reguliert wird. Diese Informationen liefern Einsicht in das komplexe Zusammenspiel der Netzwerkkomponenten einer Zelle und sind essentiell für das Verständnis der differentiellen Genexpression. Es ist zu hoffen, damit Unterschiede im Expressionsmuster der Zellen aus verschiedenen Geweben (zum Beispiel gesundes versus krankes Gewebe) aufklären zu können.
Im allgemeinen läßt sich der Reverse Engineering Prozeß in zwei Teilschritte untergliedern: Als erstes muß ein genetisches Netzwerkmodell ausgewählt werden, das zur Darstellung des Genregulationsnetzwerks dienen soll. Inwieweit man dabei von der biologischen Realität abstrahiert und welche Netzwerkkomponenten man in das Modell integriert, hängt neben der gegebenen Aufgabenstellung maßgeblich von den verfügbaren Daten ab. Der zweite Teilschritt dient anschließend dazu, die Parameter des ausgewählten Modells mit Hilfe der gegebenen Daten festzulegen und zu spezifizieren, zwischen welchen Komponenten des Netzwerkmodells regulatorische Einflüsse bestehen.
Häufig betrachtet man vereinfachend nur die regulatorischen Interaktionen während der Transkription. Es soll herausgefunden werden, wie die Transkription eines Gens durch die Produkte anderer Gene und gegebenenfalls auch durch sein eigenes Genprodukt reguliert wird. Man betrachtet dazu zum einen die entsprechenden mRNA-Konzentrationen der Gene und zum anderen die Konzentrationen der synthetisierten Proteine. Ein Reverse Engineering Algorithmus deckt dann regulatorische Zusammenhänge zwischen den gegebenen Protein- und mRNA-Konzentrationen auf und identifiziert so regulatorische Einflüsse verschiedener Genprodukte auf die Transkription der einzelnen Gene.
Viele Reverse Engineering Methoden abstrahieren sogar noch ein Stück weiter. Sie versuchen, regulatorische Beziehungen zwischen den Genen nur aus den mRNA-Konzentrationen der Gene zu identifizieren, denn diese sind, besonders seit der Ent-

wicklung der Microarray-Technologie, wesentlich leichter zu messen als die entsprechenden Protein-Konzentrationen. Aufgrund dessen sind die derzeit verfügbaren Expressionsdaten oftmals auf die mRNA-Konzentrationen der Gene beschränkt. Ignoriert man die Tatsache, daß neben der Transkriptionsregulation auch Regulationen auf anderen Ebenen der Genexpression existieren, und arbeitet mit der vereinfachten Annahme, daß aus einem Gen, ist es erst einmal transkribiert, auch ein funktionstüchtiges Genprodukt synthetisiert wird, kann man von einer starken Korrelation zwischen der mRNA-Konzentration und der Protein-Konzentration eines Gens ausgehen. Die Expressionsrate eines Gens kann deshalb allein durch seine entsprechende mRNA-Konzentration modelliert werden. Zusammenhänge zwischen den Expressionsraten zweier Gene sollen dann Rückschlüsse auf eine regulatorische Beziehung zwischen ihnen ermöglichen.
Manchmal werden zusätzlich auch extrazelluläre und intrazelluläre Einflußfaktoren (Licht, Temperatur, zugesetzte Chemikalien, Hormone, Zytokine, Nährstoffe, etc.) sowie die Abhängigkeit von Gewebe und Organ, aus dem die betrachteten Zellen stammten, in die Analyse einbezogen, um deren Einfluß auf die Expression einzelner Gene zu untersuchen.

Kapitel 2

Die Wahl eines genetischen Netzwerkmodells

Wie in Abschnitt 1.3 beschrieben, stellt sich im Reverse Engineering Prozeß zuerst die Aufgabe, ein geeignetes Netzwerkmodell auszuwählen, mit dem das zu rekonstruierende Genregulationsnetzwerk beschrieben werden soll. In [7] ist ein Modell im naturwissenschaftlichen Sinn definiert als:

Definition 2.1 (Modell [7]) *„[...] ein Abbild der Natur unter Hervorhebung für wesentlich erachteter Eigenschaften und unter Außer-Acht-Lassen als nebensächlich angesehener Aspekte. Das Modell in diesem Sinn ist ein Mittel zur Beschreibung der erfahrenen Realität [...] und Grundlage von Voraussagen über zukünftiges Verhalten des erfaßten Erfahrungsbereichs. Es ist umso realistischer oder wirklichkeitsnäher, je konsistenter es den von ihm umfaßten Erfahrungsbereich zu deuten gestattet und je genauer seine Vorhersagen zutreffen [...]“*

Ein Modell ist also eine vereinfachte und abstrahierte Darstellung eines realen Systems.
Neben der Möglichkeit, ein eigenes Netzwerkmodell zu entwerfen, bildet die Orientierung an bekannten Modellen eine Variante, ein genetisches Netzwerkmodell für das zu rekonstruierende Genregulationsnetzwerk festzulegen. In der Literatur werden eine Vielzahl möglicher Netzwerkmodelle vorgeschlagen, auf die zurückgegriffen werden kann.
Um ein zweckmäßiges Modell für die im Einzelfall zu lösende Problematik zu finden, ist es erforderlich, sich mit den an das Netzwerkmodell gestellten Anforderungen auseinanderzusetzen. Diese Anforderungen bestimmen wesentlich die Wahl eines geeigneten Netzwerkmodells, mit dem das Genregulationsnetzwerk entsprechend der Fragestellung adäquat und effizient analysiert werden kann. Daneben haben natürlich

auch die gegebenen Expressionsdaten einen entscheidenden Einfluß auf die Wahl eines Netzwerkmodells, denn diese sind sowohl bezüglich des Datenumfangs als auch in Bezug auf die experimentell verfügbaren Datentypen begrenzt.
Wichtige Eigenschaften des Netzwerkmodells, die unter Berücksichtigung der gestellten Anforderungen und der verfügbaren Expressionsdaten festzulegen sind, sollen im ersten Abschnitt dieses Kapitels näher betrachtet werden. Der zweite Abschnitt des Kapitels stellt anschließend ausgewählte genetische Netzwerkmodelle vor.

2.1 Eigenschaften eines Netzwerkmodells

In Abhängigkeit von den verfügbaren Daten und der gegebenen Fragestellung müssen bei der Wahl eines genetischen Netzwerkmodells wichtige Entscheidungen bezüglich der Eigenschaften des Netzwerkmodells getroffen werden [13]:

2.1.1 Grad der Abstraktion

Die erste wichtige Entscheidung betrifft die Genauigkeit, mit der man die Details der Genexpressions- und Genregulationsprozesse abbildet.
In einem detaillierten Modell werden die Regulationsprozesse der Genexpression sowie die Mechanismen der Transkription, der Translation und auch der posttranslationalen Modifikationen spezifiziert. Ziel ist es, die ablaufenden biochemischen Reaktionen genau zu beschreiben. Als Netzwerkkomponenten bezieht man dafür die mRNA- und Protein-Konzentrationen der Gene sowie wichtige extra- und intrazelluläre Einflußfaktoren in das Netzwerkmodell ein. Aufgrund des geringen Abstraktionsgrades können die bei der Genexpression ablaufenden Prozesse genau beschrieben werden. Allerdings ergeben sich so recht komplexe Modelle mit vielen freien Parametern, wodurch ihre Anwendung in der Regel auf sehr kleine Systeme beschränkt bleibt. Ein Beispiel für solch ein detailliertes Netzwerkmodell ist ein in [6] vorgestellter Ansatz zur Modellierung der regulatorischen Mechanismen für die Entscheidung zwischen dem lytischen und dem lysogenen Entwicklungszyklus in einem λ-Phagen. Hier werden die Mechanismen zur Regulation der Transkription und Translation auf molekularer Ebene genau modelliert und die Proteinproduktion der regulierten Gene mit Hilfe von stochastischen Kinetiken beschrieben.
Aufgrund der Komplexität biologischer Regulationsmechanismen arbeitet man aber meist mit abstrakteren Ansätzen zur Modellierung eines Genregulationsnetzwerks (vergleiche Abschnitt 1.3):
Ein erster Schritt der Abstraktion verzichtet auf eine genaue Darstellung der ablaufenden biochemischen Prozesse. In das Netzwerkmodell werden die mRNA- und Protein-Konzentrationen der Gene integriert und die Abhängigkeiten zwischen ihnen auf einer abstrakteren Ebene modelliert. Als Beispiel sei auf ein Modell in [8] verwiesen, das die Translations- und Transkriptionsprozesse mit Hilfe von Differen-

tialgleichungen beschreibt und dabei auch Zerfallsprozesse, sowohl auf der Ebene der Proteine als auch auf der Ebene der mRNA, berücksichtigt.
Ein nächster Schritt der Abstraktion betrachtet die Protein-Konzentrationen nur noch auf der Ebene ihres regulatorischen Einflusses auf die Transkription der Gene. Durch die Modellierung der Abhängigkeit der mRNA-Konzentration eines Gens von verschiedenen Protein-Konzentrationen kann die Transkriptionskontrolle dieses Gens beschrieben werden.
Aufgrund der Begrenzung derzeit verfügbarer Expressionsdaten sowohl bezüglich ihres Umfangs als auch in Bezug auf den Datentyp – oftmals werden nur die mRNA-Konzentrationen der Gene beschrieben, denn diese sind wesentlich leichter zu messen als die Protein-Konzentrationen – ist auch dieser Grad an Abstraktion häufig nicht ausreichend. Es existiert deshalb eine Vielzahl von Modellen, welche die regulatorischen Einflüsse zwischen den Genen auf einer noch abstrakteren Ebene modellieren. Vereinfachend ignorieren diese Netzwerkmodelle alle Regulationsvorgänge außerhalb der Transkription und nehmen so eine starke Korrelation zwischen den mRNA-Konzentrationen und den Proteinkonzentrationen der Gene an. Die Expressionsraten der Gene können dann allein durch die zugehörige mRNA-Konzentration beschrieben werden. Eine in einem entsprechenden Netzwerkmodell beschriebene Abhängigkeit der Expressionsrate eines Gens A von der Expressionsrate eines Gens B modelliert vereinfachend den regulatorischen Einfluß der Expression von Gen A auf die Expression von Gen B oder – noch genauer – den regulatorischen Einfluß eines aus der in Gen A verschlüsselten Information synthetisierten Proteins auf die Transkription von Gen B. Die Prozesse der Genexpression und Genregulation werden also auf einer abstrakten Ebene betrachtet, die die Details der einzelnen Prozesse völlig außer acht läßt. Extra- und intrazelluläre Einflußfaktoren kann man in das Netzwerkmodell integrieren; die Modellierung ihrer regulatorischen Einflüsse auf die Expression einzelner Gene ist aber ebenfalls nur auf der abstrakten Ebene des Netzwerkmodells möglich. Aufgrund des hohen Abstraktionsgrades ergeben sich so relativ einfache Modelle, die weniger freie Parameter besitzen und auch sehr große Systeme effizient beschreiben können.

2.1.2 Struktur versus Struktur & Dynamik

Entscheidend ist weiterhin, ob man nur die Struktur des Genregulationsnetzwerks modellieren möchte oder beides, Struktur und Dynamik.
Ein Modell der Struktur bildet lediglich die Komponenten des Genregulationsnetzwerks ab und beschreibt, zwischen welchen Komponenten regulatorische Interaktionen auftreten. Es trifft aber keine Annahmen über die Art oder Stärke einer Interaktion. Möchte man zusätzlich die Dynamik des Genregulationsnetzwerks modellieren, müssen die Interaktionen genauer spezifiziert werden. Dies erfordert weitere Entscheidungen:

2.1.3 Diskret versus kontinuierlich

Bei der Modellierung der Dynamik muß zusätzlich entschieden werden, ob mit diskreten bzw. Booleschen Variablen oder mit kontinuierlichen Variablen gearbeitet werden soll, um die einzelnen Netzwerkkomponenten zu modellieren.
Die vereinfachte Annahme von diskreten Größen führt zu weniger komplexen Modellen und erlaubt so, daß auch große genetische Netzwerke effizient analysiert werden können. Demgegenüber steht allerdings der Nachteil, daß die diskrete Beschreibung der Netzwerkkomponenten biologisch nicht realistisch ist. Eine Analyse öffentlich verfügbarer Genexpressionsdatensätze ergab, daß die gemessenen mRNA- und Protein-Konzentrationen der Gene, die ihre Expressionsraten charakterisieren, eher kontinuierliche als diskrete Größen sind. Zwar gibt es durchaus Gene, die ein sehr schnelles Schaltverhalten zwischen einem nicht exprimierenden und einem maximal exprimierenden Zustand zeigen und so die vereinfachende Annahme unterstützen, daß die entsprechenden Expressionsraten durch diskrete oder sogar Boolesche Variablen modelliert werden können. Ein Großteil der Gene nimmt aber sehr viele verschiedene Zustände an und ist vorwiegend in mittleren Stärken und nur selten überhaupt nicht oder maximal exprimiert. Bei der Diskretisierung der gemessenen Konzentrationen gehen viele Informationen verloren, denn mittlere Werte sind mit einer Booleschen Variable gar nicht und mit einer diskreten Variable nur bedingt modellierbar. Kleine und mittlere Schwankungen der mRNA- und Protein-Konzentrationen eines Gens sind deshalb nach einer Diskretisierung nicht mehr identifizierbar.

2.1.4 Deterministisch versus stochastisch

Weiterhin ist zu entscheiden, ob die regulatorischen Einflüsse zwischen den Netzwerkkomponenten durch deterministische oder stochastische Beziehungen modelliert werden sollen.
Theoretische Überlegungen und experimentelle Ergebnisse verstärken die Annahme, daß die regulatorischen Interaktionen in genetischen Netzwerken auf stochastischen Beziehungen beruhen [1, 39]. Es ist außerdem meist nicht möglich, alle bekannten Einflußgrößen genau zu messen, und aufgrund von experimentellen Meßfehlern sind die Genexpressionsdaten verrauscht, wodurch Inkonsistenzen enstehen können. Selbst wenn das zugrundeliegende System von deterministischer Natur wäre, läßt sich deshalb oft dennoch kein deterministisches Modell finden, das in der Lage ist, diese Inkonsistenzen zu modellieren. Eine stochastische Modellierung der regulatorischen Einflüsse ist damit also prinzipiell geeigneter; allerdings nimmt die Komplexität des entstehenden Modells dadurch zu.

2.2 Genetische Netzwerkmodelle

In der Literatur wird eine Vielfalt von Ansätzen zur Modellierung genetischer Netzwerke vorgeschlagen [31]. Im Rahmen der vorliegenden Arbeit sollen ausgewählte Netzwerkmodelle vorgestellt werden:

- Gerichtete Graphen
- Boolesche Netzwerke
- Diskrete dynamische Bayessche Netzwerke
- Additive Regulationsmodelle
- Kontinuierliche Dynamische Bayessche Netzwerke

Im weiteren Verlauf dieser Arbeit wird berücksichtigt, daß sich momentan verfügbare Expressionsdaten meist auf Daten aus Microarray-Experimenten zur Quantifizierung von mRNA-Konzentrationen beschränken. Dies erfordert die Verwendung eines abstrakten Netzwerkmodells, denn detaillierte Modelle mit einem geringen Abstraktionsgrad arbeiten mit Komponenten unterschiedlichen Typs und setzen die Verfügbarkeit entsprechender Daten voraus (Protein- und mRNA-Konzentrationen, genaue Ausprägung betrachteter extra- und intrazellulärer Einflußfaktoren, etc.). Damit sind die Daten aus Microarray-Experimenten zur Anpassung dieser Modelle allein nicht ausreichend. Die im folgenden vorgestellten Netzwerkmodelle arbeiten deshalb mit der vereinfachten Annahme, daß die Expressionsrate eines Gens allein durch die Konzentration seines mRNA-Transkripts beschrieben werden kann. Sie modellieren reale regulatorische Einflüsse der Expression eines Gens auf die Expression eines anderen Gens in einer abstrakten Ebene durch Abhängigkeiten zwischen den einzelnen Expressionsraten (siehe Abschnitt 2.1.1).

2.2.1 Gerichtete Graphen

Der wohl einfachste Ansatz, ein Genregulationsnetzwerk zu modellieren, ist ein gerichteter Graph. Er beschreibt lediglich die Struktur eines Netzwerks und ist wie folgt definiert:

Definition 2.2 (Gerichteter Graph) *Ein gerichteter Graph zur Modellierung eines genetischen Netzwerks ist definiert als ein Paar $\langle V, E \rangle$, wobei die Menge V von Knoten den Netzwerkkomponenten (Gene, extra- und intrazelluläre Einflußfaktoren) entspricht, und die Menge E von Kanten den regulatorischen Interaktionen zwischen ihnen.*

Somit modelliert eine Kante $A \rightarrow B$ von einem Knoten A zu einem Knoten B den regulatorischen Einfluß von einer Netzwerkkomponenten A auf eine Netzwerkkomponente B.

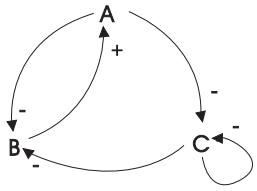


Abbildung 2.1: **Graphische Darstellung eines gerichteten Graphen.**

Mit diesem einfachen Ansatz können die einzelnen regulatorischen Einflüsse nicht näher spezifiziert werden. Es ist lediglich möglich, durch eine Beschriftung der Kanten mit „ + “ oder „ − “ anzugeben, ob es sich um einen aktivierenden oder einen inhibitorischen Einfluß handelt. Auch läßt das Modell keine Angaben über das dynamische Verhalten des Systems zu.
Die Abbildung 2.1 zeigt die graphische Darstellung eines gerichteten Graphen.

2.2.2 Boolesche Netzwerke

Zur Modellierung von Genregulationsnetzwerken wurden Boolesche Netzwerke erstmals von Stuart Kauffman [32] herangezogen. In einem Booleschen Netzwerk werden sowohl die Struktur als auch die Dynamik eines Genregulationsnetzwerks beschrieben.
Folgende formale Definition läßt sich für ein Boolesches Netzwerk als genetisches Netzwerkmodell angegeben:

Definition 2.3 (Boolesches Netzwerk) *Ein Boolesches Netzwerk ist definiert als ein Paar $\langle X, F\rangle$. Hierbei ist X eine Menge Boolescher Variablen $x_1, x_2, ..., x_N$, welche den diskretisierten Expressionsraten der Gene $g_1, g_2, ..., g_N$ entsprechen und F eine Menge von Booleschen Funktionen, die deterministische Zusammenhänge zwischen den Expressionsraten beschreiben. So gibt eine Boolesche Funktion $f_i(x_{i_1}, x_{i_2}, ..., x_{i_k})$ an, wie der Zustand (Output) der Booleschen Variable x_i (Outputelement) zum Zeitpunkt $t+1$ in Abhängigkeit von den Zuständen (Input) der Booleschen Variablen $x_{i_1}, x_{i_2}, ..., x_{i_k}$ (Inputelemente) zum Zeitpunkt t bestimmt wird. Sie modelliert damit die regulatorischen Einflüsse auf die Expression von Gen g_i.*

Es wird also die vereinfachende Annahme getroffen, daß ein Gen sich entweder in einem aktiven Zustand befindet, in dem es exprimiert wird, oder in einem inaktiven Zustand, in dem es nicht oder nur sehr wenig exprimiert wird. Folglich läßt sich die Expressionsrate eines Gens g_i auf einer qualitativen Ebene durch eine Boolesche Variable x_i modellieren. Der Zustand eines Booleschen Netzwerks zu einem Zeitpunkt t wird durch einen Vektor der Dimension N angegeben; der Zustandsraum eines Booleschen Netzwerks besteht aus 2^N globalen Zuständen.

t	1	2	3	4
x_1	0	1	0	0
x_2	1	0	0	0
x_3	0	0	0	0

(a)

t	1	2	3	4	5
x_1	0	1	1	1	1
x_2	1	1	0	1	0
x_3	1	0	1	0	1

(b)

Tabelle 2.1: **Trajektorien eines Booleschen Netzwerks.** Die Trajektorie in (a) befindet sich ab dem Zeitpunkt $t = 3$ in einem stabilen Attraktorzustand; die Trajektorie in Tabelle (b) ab dem Zeitpunkt $t = 2$ in einem zyklischen Attraktorzustand.

Bezüglich der Dynamik des Systems arbeitet man vereinfachend mit einem diskreten Zeitsystem. Ein deterministischer Zustandsübergang vom Zustand des Netzwerks zu einem Zeitpunkt t in den Zustand des Netzwerks zum Zeitpunkt $t + 1$ wird mit Hilfe von Booleschen Funktionen modelliert; die Aktualisierung der Zustände aller Variablen x_i erfolgt synchron. Die zeitliche Entwicklung eines genetischen Netzwerks nennt man eine Trajektorie:

Definition 2.4 (Trajektorie) *Eine Trajektorie ist eine aus einer Folge von Zustandsübergängen resultierende Sequenz von globalen Systemzuständen.*

Erreicht ein genetisches Netzwerk im Verlauf der zeitlichen Entwicklung einen Systemzustand, den es zu einem vorangegangenen Zeitpunkt bereits einmal angenommen hatte, ist es in einem Attraktor angelangt.

Definition 2.5 (Attraktor) *Nimmt das System einen globalen Zustand an, der zuvor bereits einmal aufgetreten ist, befindet sich das System in einem stationären Zustand bzw. stabilen Zyklus, den man als Attraktor bezeichnet.*

Da es speziell in einem Booleschen Netzwerk nur eine begrenzte Anzahl von globalen Systemzuständen gibt, führt hier jeder beliebige Startzustand einer Trajektorie nach endlich vielen Zustandsübergängen in einen Attraktor. In Tabelle 2.1 sind zwei verschiedene Trajektorien eines Booleschen Netzes mit den drei Netzwerkkomponenten x_1, x_2 und x_3 dargestellt. Das entsprechende System befindet sich in 2.1 (a) ab dem Zeitpunkt $t = 3$ in einem stabilen und in 2.1 (b) ab dem Zeitpunkt $t = 2$ in einem zyklischen Attraktorzustand. Ein Boolesches Netzwerk in einem Attraktorzustand kann als Genregulationsnetzwerk einer stabilen, differenzierten Zelle interpretiert werden [33].
Prinzipiell gibt es 2^{2^k} mögliche Boolesche Funktionen $f_i(x_{i_1}, x_{i_2}, ...x_{i_k})$, mit denen die Abhängigkeit des Outputelements x_i von den k Inputelementen $x_{i1}, x_{i2}, ..., x_{ik}$ modelliert werden kann. Mit der Begründung, daß nur eine Teilmenge dieser 2^{2^k} möglichen Booleschen Funktionen sich chemisch einfach realisieren läßt und so biologisch re-

levant erscheint, wird in [33] vorgeschlagen, die Menge der möglichen Booleschen Funktionen auf diese Teilmenge der kanalisierenden Funktionen (*engl.: canalyzing function*) zu beschränken, die wie folgt definiert sind:

Definition 2.6 (Kanalisierende Funktion) *Eine Boolesche Funktion f ist eine kanalisierende Funktion, falls sich für mindestens eines ihrer Inputelemente (kanalisierendes Inputelement) x_j zwei Zustände $u, v \in 0, 1$ finden lassen, so daß die Implikation gilt: $x_j = u \rightarrow f = v$. Der Zustand des Outputelements der Booleschen Funktion f wird also durch mindestens einen Zustand des Inputelements x_j unabhängig von den Zuständen aller anderen Inputelemente garantiert.*

Ein Beispiel für eine kanalisierende Funktion ist die ODER Funktion $f(x_1, x_2) = x_1 \vee x_2$. Sowohl x_1 als auch x_2 sind hier kanalisierende Inputelemente, denn für den Zustand „1“ garantieren sie unabhängig voneinander auch den Zustand „1“ des Outputelements.
Für die grafische Darstellung eines Booleschen Netzwerks werden mehrere Möglichkeiten vorgeschlagen (Abbildung 2.2). So lassen sich einerseits eine logische Schaltung (Abbildung 2.2 (a)) oder auch ein Wiring Diagramm (Abbildung 2.2 (d)) verwenden. Andererseits ist es auch möglich, entsprechende Regeltabellen (Abbildung 2.2 (c)) oder eine Liste der Booleschen Funktionen (Abbildung 2.2 (b)) anzugeben.
Die Integration intra- und extrazellulärer Einflußfaktoren ist hier nur bedingt möglich. Die Ausprägungen der betrachteten Faktoren müssen dafür durch Boolesche Variablen e_k beschrieben und in den Booleschen Funktionen f_i der von ihnen abhängigen Expressionsraten x_i entsprechend berücksichtigt werden. Für viele Einflußfaktoren macht die Abbildung auf eine Boolesche Variable durchaus Sinn; sie modelliert mit den Zuständen „1“ und „0“ die jeweilige An- bzw. Abwesenheit des betreffenden Faktors. Andere Faktoren, wie zum Beispiel die Temperatur oder das Gewebe, aus dem die betrachtete Zelle stammt, können durch eine Boolesche Variable nur schwer erfaßt werden.

2.2.3 Diskrete Dynamische Bayessche Netzwerke (diskrete DBN)

Ein Bayessches Netzwerk [48] ist ein Beispiel für ein Netzwerkmodell, das Interaktionen zwischen den Netzwerkkomponenten durch stochastische Beziehungen repräsentiert. Bayessche Netzwerke finden eine breite Anwendung im Bereich der künstlichen Intelligenz, wo sie zur Modellierung von Unsicherheit benutzt werden. Für die Identifizierung von Genregulationsnetzwerken aus Genexpressionsdaten wurden sie erstmals von Friedman *et al.* [18] genutzt. Ähnlich wie bei den Booleschen Netzwerken verwendet man ebenfalls diskretisierte Expressionsraten. Im Unterschied

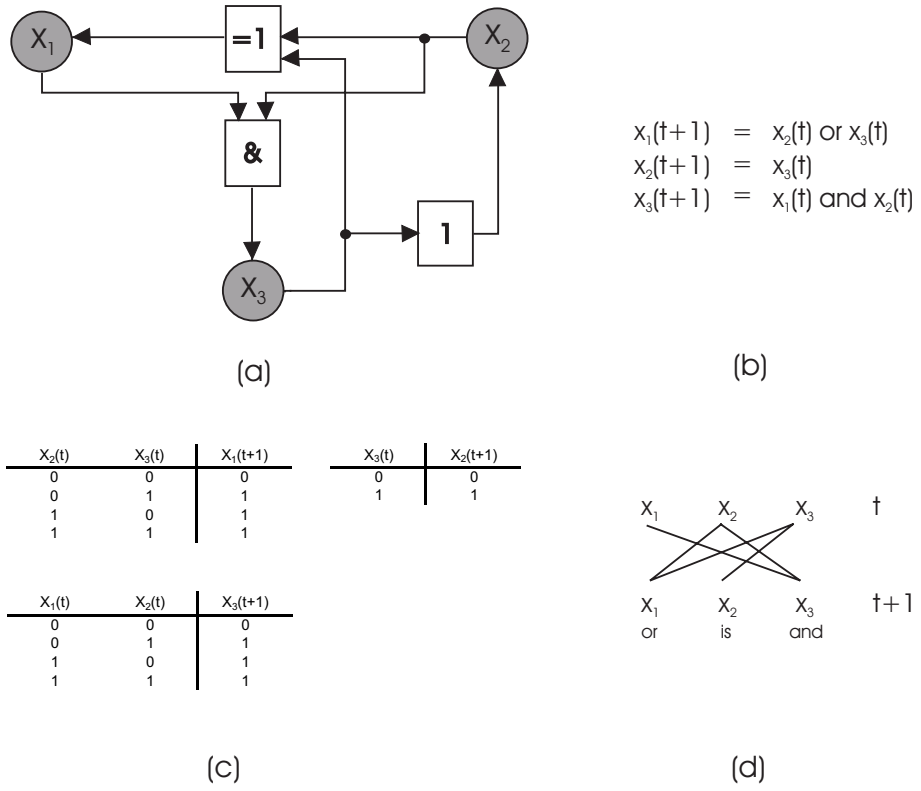


Abbildung 2.2: **Darstellungsmöglichkeiten eines Booleschen Netzwerks:** (a) logischer Schaltplan, (b) Liste der Booleschen Funktionen, (c) Regeltabellen, (d) Wiring Diagramm.

zum Booleschen Ansatz, der die Genexpression in der Regel nur auf einer qualitativen Ebene betrachtet, arbeitet man hier aber auf einer quantitativen Ebene, und ein Gen darf mehr als die zwei Zustände „1“ (an) und „0“ (aus) annehmen – oft werden in einem diskreten Bayesschen Netzwerk die drei Zuständen „-1“ (die Expressionsrate eines Gens ist signifikant kleiner als seine mittlere Expressionsrate), „0“ (die Expressionsrate eines Gens entspricht ungefähr seiner mittleren Expressionsrate) und „1“ (die Expressionsrate eines Gens ist signifikant größer als seine mittlere Expressionsrate) verwendet. Mit diesem Ansatz können also auch mittlere Expressionsraten modelliert werden.
Die Prozesse der Genexpression erscheinen aufgrund von Meßfehlern und fehlenden wichtigen Einflußfaktoren oft stochastisch, und es ist deshalb nicht möglich, deterministische Einflüsse zwischen den Netzwerkkomponenten zu identifizieren. Durch ihre wahrscheinlichkeitstheoretische Natur sind Bayessche Netzwerke im Vergleich zu den Booleschen Netzwerken besser in der Lage, mit Meßfehlern und Inkonsistenzen in den Expressionsdaten umzugehen.
Ein klassisches Bayessches Netzwerk wird durch eine gerichtete Graphenstruktur G und eine Menge Θ von Parametern definiert. Im Gegensatz zum Booleschen An-

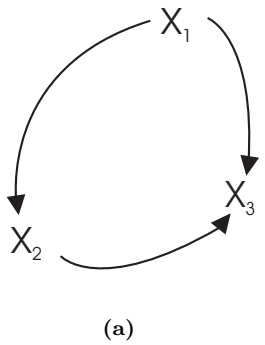
satz sind hier nur azyklische Graphenstrukturen zugelassen. Die Menge der Knoten von G entspricht einer Menge von diskreten Zufallsvariablen $\mathbf{X} = \{X_1, X_2, ...X_N\}$, die Kanten beschreiben stochastische Relationen zwischen den Zufallsvariablen. Existiert eine gerichtete Kante $X_j \rightarrow X_i$ von einem Knoten X_j zu einem Knoten X_i, dann wird X_j „Elternknoten von X_i“ und X_i „Kindsknoten von X_j“ genannt und X_j gehört so zu der Elternmenge $\mathrm{Pa}(X_i)$ von X_i. Gegeben seine Eltern ist jeder Knoten X_i unabhängig von allen anderen Knoten [16]:

$$P(X_i|X_1, X_2, ..., X_N) = P(X_i|Pa(X_i)) \tag{2.1}$$

Diese Eigenschaft nennt man auch bedingte Unabhängigkeit.
Die Graphenstruktur G repräsentiert also die Beziehungen zwischen den Zufallsvariablen auf einer qualitativen Ebene in Form von bedingten Unabhängigkeitsrelationen. Die Parametermenge Θ hingegen definiert für jede Zufallsvariable eine diskrete bedingte Wahrscheinlichkeitsverteilung und beschreibt so die Beziehungen zwischen ihnen auf einer quantitativen Ebene. Konkret enthält sie für jede Kombination der möglichen Werte x_i einer Zufallsvariablen X_i mit den möglichen Werten pa_i ihrer Eltern $Pa(X_i)$ den Parameter $\theta_{i,x_i,pa_i} = P(X_i = x_i|Pa(X_i) = pa_i)$. Dieser gibt an, mit welcher Wahrscheinlichkeit die Zufallsvariable X_i den Wert x_i annimmt, unter der Bedingung, daß ihre Eltern $Pa(X_i)$ die Werte pa_i angenommen haben. Diese bedingte Wahrscheinlichkeit ist definiert durch :

$$P(X_i = x_i|Pa(X_i) = pa_i) = \frac{P(X_i = x_i, Pa(X_i) = pa_i)}{P(Pa(X_i) = pa_i)} \tag{2.2}$$

Die diskrete bedingte Wahrscheinlichkeitsverteilung einer Zufallsvariablen X_i kann in einer bedingten Wahrscheinlichkeitstabelle angegeben werden, die für jede Kombination der möglichen Werte x_i der Zufallsvariable X_i und der möglichen Werte pa_i ihrer Eltern $Pa(X_i)$ den jeweiligen Parameter θ_{i,x_i,pa_i} enthält.
Ein Beispiel für die graphische Darstellung eines klassischen Bayesschen Netzwerks findet man in Abbildung 2.3. Es sind sowohl die Graphenstruktur G (Abbildung 2.3 (a)) als auch die einzelnen bedingten Wahrscheinlichkeitstabellen für die Knoten (Abbildung 2.3 (b)-(d)) angegeben.
Aus der Graphenstruktur G und der Parametermenge Θ ergibt sich die Verbundwahrscheinlichkeitsverteilung für das gesamte Netzwerk, die jedem einzelnen Systemzustand $\mathbf{x} = \{x_1, x_2, ..., x_N\}$ eine Wahrscheinlichkeit $P(\mathbf{X} = \mathbf{x}) = P(X_1 = x_1, X_2 = x_2, ..., X_N = x_N)$ für sein Auftreten zuordnet. Unter der Annahme, daß die Reihenfolge der Zufallsvariablen konsistent mit der azyklischen Graphenstruktur ist (d.h. es gilt: $X_i \in Pa(X_j) \rightarrow i < j$), kann diese Wahrscheinlichkeit aus dem Produkt der bedingten Wahrscheinlichkeiten für die einzelnen Zufallsvariablen des Netzwerkes berechnet werden:



(a)

(b)

$P(X_1 = 0)$	$P(X_1 = 1)$
0.99	0.01

(c)

X_1	$P(X_2 = 0\|X_1)$	$P(X_2 = 1\|X_1)$
0	0.9	0.1
1	0.15	0.85

(d)

X_1	X_2	$P(X_3 = 0\|X_1, X_2)$	$P(X_3 = 1\|X_1, X_2)$
0	0	0.9	0.1
0	1	0.15	0.85
1	0	0.15	0.85
1	1	0.05	0.95

Abbildung 2.3: **Graphische Darstellung eines klassischen Bayesschen Netzwerks:** In (a) ist die Graphenstruktur G beschrieben; (b),(c) und (d) geben die einzelnen bedingten Wahrscheinlichkeitstabellen an, die die bedingten Wahrscheinlichkeitsverteilungen der Knoten X_1, X_2 und X_3 beschreiben.

$$
\begin{aligned}
& P(X_1 = x_1, X_2 = x_2, ..., X_N = x_N) \\
= \; & P(X_N = x_N | X_1 = x_1, X_2 = x_2, ..., X_{N-1} = x_{N-1}) \cdot \\
& P(X_1 = x_1, X_2 = x_2, ..., X_{N-1} = x_{N-1}) \\
= \; & P(X_N = x_N | X_1 = x_1, X_2 = x_2, ..., X_{N-1} = x_{N-1}) \cdot \\
& P(X_{N-1} = x_{N-1} | X_1 = x_1, X_2 = x_2, ..., X_{N-2} = x_{N-2}) \cdot \\
& P(X_1 = x_1, X_2 = x_2, ..., X_{N-2} = x_{N-2}) \\
= \; & \cdots \\
= \; & \prod_{i=0}^{N} P(X_i = x_i | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) \qquad (2.3)
\end{aligned}
$$

Unter Berücksichtigung der bedingten Unabhängigkeit (Eigenschaft 2.1), kann man die einzelnen bedingten Wahrscheinlichkeiten weiter vereinfachen:

$$
\begin{aligned}
& \prod_{i=0}^{N} P(X_i = x_i | X_1 = x_1, X_2 = x_2, ..., X_{i-1} = x_{i-1}) \\
= \; & \prod_{i=0}^{N} P(X_i = x_i | Pa(X_i) = pa_i) \qquad (2.4)
\end{aligned}
$$

Übertragen auf ein Bayessches Netzwerkmodell für ein Genregulationsnetzwerk entsprechen die stochastischen Zufallsvariablen den Expressionsraten der Gene. Eine

Kante $X_j \rightarrow X_i$ zwischen zwei Zufallsvariablen X_j und X_i beschreibt einen stochastischen Einfluß der Expressionsrate von Gen g_j auf die Expressionsrate von Gen g_i und modelliert damit auf einer abstrakten Ebene den regulatorischen Einfluß der Expression von Gen g_j auf die Expression von Gen g_i. Die in Θ für eine Zufallsvariable definierte bedingte Wahrscheinlichkeitsverteilung beschreibt die Verteilung der Expressionsrate des zugehörigen Gens in Abhängigkeit der Expressionsraten seiner regulierenden Gene. Intra- und extrazelluläre Einflußfaktoren können durch entsprechende diskrete Variablen dargestellt werden. Ihre Ausprägungen muß man dazu gegebenenfalls diskretisieren. Es sind sowohl quantitative Merkmale (zum Beispiel Konzentration einer zugesetzten Chemikalie) als auch qualitative Merkmale (zum Beispiel der Gewebetyp der Zelle) darstellbar. Durch das Einfügen der entsprechenden diskreten Variable in die Elternmenge $Pa(X_i)$ kann dann der regulatorische Einfluß eines Faktors auf die Expression des Gens g_i modelliert werden.

Im Gegensatz zu den klassischen Bayesschen Netzwerken, die keine zeitliche Evolution beschreiben können, beziehen Dynamische Bayessche Netzwerke die Zeit als zusätzliche Dimension ein. Wie bei den Booleschen Netzwerken arbeitet man mit einem diskreten Zeitsystem, in dem die Zustände der Knoten synchron aktualisiert werden. Die Knotenmenge der Graphenstruktur G entspricht nun der Menge $\mathbf{X}$ von stochastischen Zufallsvariablen $\mathbf{X} = \{\mathbf{X}[0], \mathbf{X}[1], ..., \mathbf{X}[T]\} = \{X_1[0], X_2[0], ..., X_N[0], X_1[1], X_2[1], ..., X_N[1], ..., X_1[T], X_2[T], ..., X_N[T]\}$, welche – übertragen auf die Domäne der Genregulationsnetzwerke – die diskretisierten Expressionsraten der Gene $g_1, g_2, ..., g_N$ zu den jeweiligen Zeitpunkten $0, 1, ..., T$ modellieren. Die Expressionsrate x_i von Gen g_i zu einem bestimmten Zeitpunkt t wird also durch die Zufallsvariable $X_i[t]$ beschrieben; der Systemzustand des Netzwerks zum Zeitpunkt t kann durch die Teilmenge $\mathbf{X}[t] = \{X_1[t], X_2[t], ..., X_N[t]\}$ angegeben werden. Zusammenfassend ergibt sich damit folgende formale Definition für ein diskretes Dynamisches Bayessches Netzwerk als Modell für ein Genregulationsnetzwerk:

Definition 2.7 (Diskretes DBN) *Ein diskretes Dynamisches Bayessches Netzwerk ist definiert als Paar $\langle G, \Theta \rangle$. G ist ein azyklischer, gerichteter Graph, dessen Knotenmenge der Menge von diskreten Zufallsvariablen $\mathbf{X} = \{\mathbf{X}[0], \mathbf{X}[1], ..., \mathbf{X}[T-1]\} = \{X_1[0], X_2[0], ..., X_N[0], X_1[1], X_2[1], ..., X_N[1], ..., X_1[T], X_2[T], ..., X_N[T]\}$ entspricht, welche die diskretisierten Expressionsraten der Gene $g_1, g_2, ..., g_N$ zu den Zeitpunkten $0, 1, ..., T$ beschreiben. Eine Kante zwischen zwei Knoten $X_i[t_m] \rightarrow X_j[t_n]$ ($t_m \leq t_n$) zeigt an, daß die Expressionsrate des Gens g_i zum Zeitpunkt t_m die Expressionsrate des Gens g_j zum Zeitpunkt t_n direkt beeinflußt und damit $X_i[t_m]$ zu der Menge $Pa(X_j[t_n])$ der Eltern von $X_j[t_n]$ gehört. Für jede Kombination der möglichen Werte x_i einer Zufallsvariable $X_i[t]$ und der möglichen Werte pa_i ihrer Eltern $Pa(X_i[t])$ enthält die Parametermenge Θ den Parameter $\theta_{i,x_i,pa_i} = P(X_i[t] = x_i | Pa(X_i[t]) = pa_i)$. Damit definiert Θ für jede diskrete Zufallsvariable $X_i[t]$ ihre zugehörige diskrete bedingte Wahrscheinlichkeitsverteilung.*

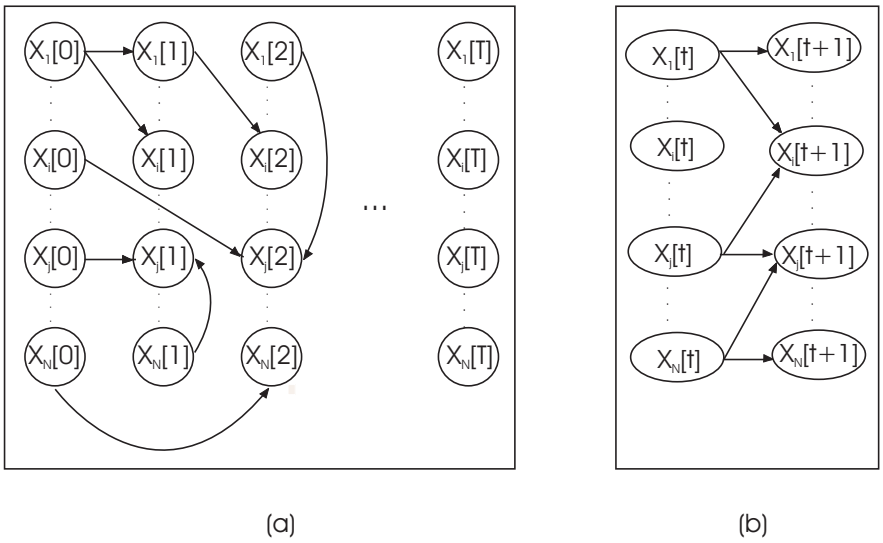


Abbildung 2.4: **Beispiel für eine Graphenstruktur eines Dynamischen Bayesschen Netzwerks.** Übertragen auf ein Genregulationsnetzwerk repräsentieren die Knoten die Expressionsraten der Gene zu einem bestimmten Zeitpunkt; die Kanten beschreiben stochastische Interaktionen zwischen ihnen und modellieren so reale regulatorische Einflüsse. Im Unterschied zu (a) wird in (b) angenommen, daß es sich bei den dynamischen Prozessen in einem Genregulationsnetzwerk um stationäre Markov-Prozesse handelt.

Oftmals wird angenommen, daß es sich bei den dynamischen Prozessen in einem Genregulationsnetzwerk um stationäre Markov-Prozesse handelt und damit gilt:

1. $P(\mathbf{X}[t+1]|\mathbf{X}[t], \mathbf{X}[t-1], ..., \mathbf{X}[0]) = P(\mathbf{X}[t+1]|\mathbf{X}[t])$
2. $P(\mathbf{X}[t+1]|\mathbf{X}[t])$ ist unabhängig von t

Es muß daher lediglich ein Zustandsübergang des Netzes von einem Zeitpunkt t zu einem Zeitpunkt $t+1$ modelliert werden. Das Netzwerkmodell enthält so nur die diskreten Zufallsvariablen $X_1[t], X_2[t], ..., X_N[t]$, welche die Expressionsraten der Gene $g_1, g_2, .., g_N$ vor einem Zustandsübergang beschreiben, und die diskreten Zufallsvariablen $X_1[t+1], X_2[t+1], ..., X_N[t+1]$, die die Expressionsraten der Gene $g_1, g_2, .., g_N$ nach einem Zustandsübergang charakterisieren.
Zwei Beispiele für eine Graphenstruktur G eines diskreten DBN sind in Abbildung 2.4 dargestellt.

2.2.4 Additive Regulationsmodelle

Die Expressionsrate eines Gens ist vielmehr eine kontinuierliche als eine diskrete Größe. Diese Tatsache lieferte die Motivation für die Entwicklung eines Netzwerkmodells, welches – im Gegensatz zu den vorangehenden Netzwerkmodellen – die

Expressionsraten mit kontinuierlichen Variablen modelliert. In der Literatur sind viele ähnliche Vorschläge für ein solches Netzwerkmodell unter unterschiedlichen Namen zu finden: „Konnektionistisches Modell“ (*engl.: connectionist model*) in [44], „Lineares Modell“ (*engl.: linear model*) (linearer Ansatz) und „Modellierung mit rekurrenten neuronalen Netzwerken“ (*engl.: modelling with recurrent neural networks*) (nichtlinearer Ansatz) in [13], „Lineares Transkriptionsmodell“ (*engl.: linear transcription model*) in [8] und „Gewichtsmatrixmodell“ (*engl.: weight matrix model*) in [64]. All diese Varianten beruhen auf der vereinfachenden Annahme, daß die regulatorischen Einflüsse auf ein zu regulierendes Gen unabhängig voneinander sind und ihre Wirkung additiv ist. Deshalb wird in [12] vorgeschlagen, die einzelnen Varianten unter dem Namen „Additive Regulationsmodelle“ (*engl.: additive regulation models*) zusammenzufassen. Es kann die folgende formale Definition angegeben werden:

Definition 2.8 (Additives Regulationsmodell) *Ein Additives Regulationsmodell definiert ein genetisches Netzwerk durch das Tripel $\langle X, W, E\rangle$. X ist eine Menge von kontinuierlichen Variablen $x_1, x_2, ..., x_N$, die die kontinuierlichen Expressionsraten der Gene $g_1, g_2, ..., g_N$ beschreiben, W ist eine Gewichtsmatrix und E ist eine Menge von Differentialgleichungen. Die Einträge der Gewichtsmatrix W spezifizieren die regulatorischen Interaktionen zwischen den Genen. Für jede Variable x_i enthält E eine Differentialgleichung, welche deterministisch die zeitliche Änderung von x_i in Abhängigkeit von allen anderen Variablen beschreibt. Möglich sind sowohl lineare als auch nichtlineare Differentialgleichungen.*

Den Kern dieses Ansatzes bildet die Gewichtsmatrix W zur Spezifizierung der in dem Genregulationsnetzwerk auftretenden regulatorischen Interaktionen. Der regulatorische Einfluß der Expression eines Gens g_j auf die Expression eines Gen g_i wird durch das Gewicht w_{ij} näher spezifiziert – der Betrag $|w_{ij}|$ gibt die Stärke des Einflusses an und sein Vorzeichen, ob es sich um einen aktivierenden oder einen inhibitorischen Einfluß handelt. Hat das Gewicht w_{ij} den Wert 0, dann ist die Expression von Gen g_i unabhängig von der Expression des Gens g_j. Mit Hilfe einer Differentialgleichung wird die zeitliche Änderung der Expressionsrate in Abhängigkeit von den, jeweils mit dem zugehörigen Gewicht bewerteten, Expressionsraten aller anderen Gene beschrieben. In dem einfacheren linearen Ansatz (Abbildung 2.5 (a)) hängt die Expressionsrate x_i des Gens g_i linear von ihrem regulatorischen Input r_i ab, der als Summe über die gewichteten Expressionsraten aller Gene definiert ist:

$$\frac{dx_i}{dt} = r_i + \beta_i - D_i x_i = \sum_{j=1}^{N} w_{ij} x_j + \beta_i - D_i x_i \tag{2.5}$$

Die genspezifische Konstante β_i ist ein sogenannter Biasfaktor, der das Expressionsverhalten von Gen g_i bestimmt, falls kein regulatorischer Input vorliegt ($r_i = 0$). Die Zerfallskonstante D_i definiert die Rate, mit der das Produkt von Gen g_i abgebaut

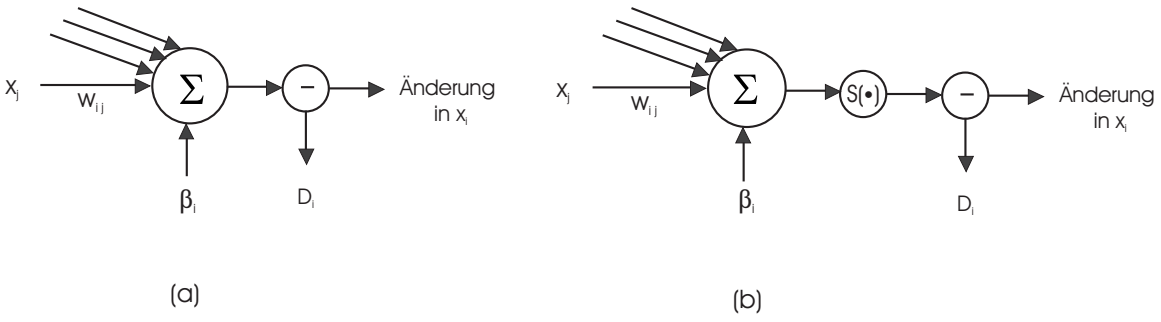


Abbildung 2.5: **Schematische Darstellung des Additiven Regulationsmodells [13]:** (a) linearer Ansatz; (b) nichtlinearer Ansatz

wird. Diese Differentialgleichung (2.5), welche die Änderung der Expressionsrate eines Gens in einem kontinuierlichen Zeitsystem beschreibt, kann durch die Einführung von diskreten Zeitschritten Δt in eine Differenzengleichung überführt werden:

$$\begin{aligned} \frac{\Delta x_i}{\Delta t} &= \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} \\ &= \sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i - D_i x_i \end{aligned} \tag{2.6}$$

Durch die Multiplikation mit Δt und die Addition von $x_i(t)$ ergibt sich die zu 2.6 äquivalente Aktualisierungsregel (*engl.: update rule*):

$$\begin{aligned} x_i(t+\Delta t) &= \sum_{j=1}^{N} w_{ij} x_j(t)\Delta t + x_i(t) + \beta_i \Delta t - D_i x_i \Delta t \\ &= \sum_{j=1}^{N} w'_{ij} x_j(t) + \beta'_i \end{aligned} \tag{2.7}$$

Wobei gilt: $w'_{ij} = \Delta t w_{ij} + (1 - D_i \Delta t)\delta_{ij}$ und $\beta'_i = \Delta t \beta_i$. Dabei ist δ_{ij} das Kronecker Symbol, welches den Wert 1 annimmt, falls $i = j$.
Vereinfachend arbeitet man mit $\Delta t = 1$ und $D_i = 1$:

$$x_i(t+1) = \sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i \tag{2.8}$$

Dieser lineare Ansatz hat zwei wesentliche Nachteile. Zum einen kann ein solches lineares, additives System nur einen Attraktor im Zustandsraum haben [13]. Interpretiert man, wie in [33] vorgeschlagen, Attraktoren eines Genregulationsnetzwerks

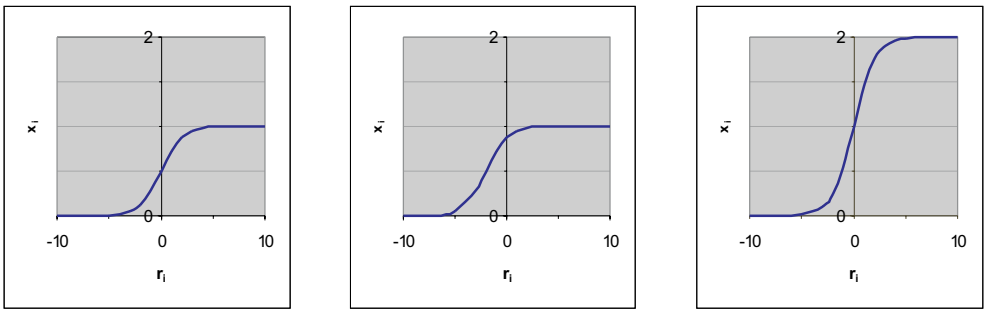


Abbildung 2.6: **Abhängigkeit der Sigmoidalfunktion von den Parametern β_i und max_i:** (a) $\beta_i = 0$, $max_i = 1$; (b) $\beta_i = 2$, $max_i = 1$;(c) $\beta_i = 0$, $max_i = 2$.

als stabile Zelltypen mit einer jeweils charakteristischen Morphologie und spezifischen Funktion, dann können mit diesem Ansatz nur Zellen mit einem einzigen stabilen Zelltyp modelliert werden. Ein weiterer Nachteil des linearen Ansatzes ergibt sich daraus, daß die Expressionsrate eines Gens weder nach oben, noch nach unten beschränkt ist. Sie kann also einerseits beliebig groß werden, andererseits aber auch negative Werte annehmen. Beides ist biologisch unrealistisch.
Abhilfe kann durch die Einführung einer sigmoiden Funktion $S(\cdot)$ geschaffen werden, die einen nichtlinearen Zusammenhang zwischen dem regulatorischen Input r_i eines Gens g_i und seiner Expressionsrate x_i erzeugt (Abbildung 2.5 (b)):

$$\begin{aligned} \frac{dx_i}{dt} &= S(r_i + \beta_i) - D_i x_i \\ \frac{dx_i}{dt} &= S(\sum_{j=1}^{N} w_{ij} x_j + \beta_i) - D_i x_i \\ &= \frac{max_i}{1 + e^{-(\sum_{j=1}^{N} w_{ij} x_j + \beta_i)}} - D_i x_i \end{aligned} \tag{2.9}$$

Hierbei beschreibt die genspezifische Konstante max_i die maximale Expressionsrate des Gens g_i; D_i definiert wieder die Zerfallskonstante. Analog zum linearen Ansatz legt der genspezifische Biasfaktor β_i das Expressionsverhalten von Gen g_i im Falle eines fehlenden regulatorischen Inputs ($r_i = 0$) fest. Wie zu erkennen, wird die Expressionsrate eines Gens in diesem nichtlinearen Ansatz auf das Intervall $(0, max_i)$ beschränkt (Abbildung 2.6).
Äquivalent zum linearen Ansatz kann die Differentialgleichung 2.9 durch die Einführung von diskreten Zeitschritten Δt und den Annahmen $D_i = 1$ sowie $\Delta t = 1$ in

eine Differenzengleichung (2.10) und in die zugehörige Aktualisierungsregel (2.11) umgeformt werden:

$$\begin{aligned} \frac{\Delta x_i}{\Delta t} &= \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} \\ &= \frac{max_i}{1 + e^{-(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i)}} - D_i x_i(t) \end{aligned} \tag{2.10}$$

$$\begin{aligned} x_i(t+\Delta t) &= \frac{max_i}{1 + e^{-(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i)}} \Delta t - D_i x_i(t) \Delta t + x_i(t) \\ x_i(t+1) &= \frac{max_i}{1 + e^{-(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i)}} \end{aligned} \tag{2.11}$$

Der Einfluß extra- und intrazellulärer Faktoren auf die Expression eines Gens g_i kann in dieser abstrakten Ebene additiver Einflüsse ebenfalls in das Modell integriert werden. Notwendig werden dafür diskrete bzw. kontinuierliche Variablen e_k, die die Ausprägung der betrachteten Faktoren beschreiben, und entsprechende Gewichte w_{ie_k}, die den jeweiligen Einfluß näher spezifizieren.

2.2.5 Kontinuierliche Dynamische Bayessche Netzwerke (kontinuierliche DBN)

Wie in Abschnitt 2.2.3 beschrieben, arbeiten diskrete (Dynamische) Bayessche Netzwerke mit diskreten Zufallsvariablen, um die eigentlich kontinuierlichen Expressionsraten der Gene zu modellieren. Werden bei der Diskretisierung die kontinuierlichen Expressionsraten der Gene auf nur wenige diskrete Werte abgebildet, so geht dadurch – wie bereits mehrfach erwähnt – sehr viel Information verloren. Verwendet man hingegen eine sehr feine Quantifizierung, enthält das resultierende diskrete Netzwerkmodell zuviele Parameter.
In der Literatur finden sich aber auch Ansätze, in denen (Dynamische) Bayessche Netze mit kontinuierlichen Variablen studiert werden [19, 23, 24, 28].
Im Gegensatz zu den diskreten (Dynamischen) Bayesschen Netzwerken entsprechen die Knoten von G in einem kontinuierlichen (Dynamischen) Bayesschen Netzwerk $B = \langle G, \Theta \rangle$ kontinuierlichen Zufallsvariablen und modellieren so die kontinuierlichen Expressionsraten der Gene. In Θ ist für jeden Knoten X_i eine kontinuierliche bedingte Wahrscheinlichkeitsdichte $f(x_i|pa_i)$ definiert, welche die Wahrscheinlichkeiten $P(X_i = x_i|Pa(X_i) = pa_i) = P(X_i = x_i|X_1 = x_1, X_2 = x_2, ...X_k = x_k)$ für die einzelnen Werte von X_i in Abhängigkeit von seinen Eltern $X_1, X_2, ..., X_k$ beschreibt. Dies geschieht hier durch die Angabe der Parameter der Wahrscheinlichkeitsverteilung.

Für die Wahl einer solchen kontinuierlichen bedingten Wahrscheinlichkeitsverteilung gibt es mehrere Möglichkeiten. Die einfachste Variante besteht darin, eine Normalverteilung zu verwenden [19]:

$$X_i \sim N(a_0 + \sum_{j=1}^{k} a_j \cdot x_j, \sigma^2) \tag{2.12}$$

oder genauer

$$\begin{aligned} & f(x_i|pa_i) \\ = \; & f(x_i|x_1, x_2, ..., x_k) \\ = \; & \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{\left(x_i - (a_0 + \sum_{j=1}^{k} a_j \cdot x_j)\right)^2}{2\sigma^2} \right) \end{aligned} \tag{2.13}$$

Kontinuierliche (Dynamische) Bayessche Netzwerke, die mit diesen bedingten Wahrscheinlichkeitsverteilungen arbeiten, sind auch unter dem Namen Gauß'sche Netzwerke bekannt und wurden ausführlich von Heckerman *et al.* [23] studiert.
Ein Nachteil dieser Gauß'schen Netzwerke ist, daß sie nur lineare Abhängigkeiten zwischen den einzelnen Variablen modellieren können. In [19, 20, 65] wird deshalb vorgeschlagen, die Wahrscheinlichkeitsverteilung eines Knotens X_i durch eine Gauß'sche Mischverteilung aus einer bestimmten Anzahl L_i von Normalverteilungen zu beschreiben, um so diese Einschränkung zu beheben [19]:

$$P(X_i = x_i|Pa(X_i) = pa_i) = \sum_{l=1}^{L_i} w_l f_l\left(x_i|pa_i\right) \tag{2.14}$$

Hierbei beschreibt jedes f_l eine Normalverteilung; die bedingte Wahrscheinlichkeitsverteilung für X_i ergibt sich aus der gewichteten Summe der einzelnen f_l.
Ein weiterer Ansatz läßt sich in [28]finden. Um ein Bayessches Netz mit nichtlinearen Interaktionen zwischen den kontinuierlichen Variablen aus gegebenen Trainingsdaten zu erlernen, werden hier nichtparametrische Dichtefunktionen benutzt. Grob beschrieben, kann die Wahrscheinlichkeitsverteilung des Knotens X_i zu den gegebenen Trainingsdaten $x_i[1], x_i[2], ..., x_i[M]$ durch die folgende Gleichung geschätzt werden:

$$P(X_i = x_i) = \frac{1}{M} \sum_{m=l}^{M} g\left(\frac{1}{\sigma} \parallel x_i - x_i[m] \parallel_2\right) \tag{2.15}$$

Hierbei repräsentieren $g(\cdot)$ die Dichte der Standardnormalverteilung und σ einen Glättungsparameter. Analog läßt sich die Wahrscheinlichkeitsverteilung $P(X_i =$

$x_i, Pa(X_i) = pa_i)$ bestimmen. Die bedingte Wahrscheinlichkeitsverteilung von Knoten X_i gegeben seine Eltern $Pa(X_i)$ ergibt sich dann aus:

$$P(X_i = x_i | Pa(X_i) = pa_i) = \frac{P(X_i = x_i, Pa(X_i) = pa_i)}{P(Pa(X_i) = pa_i)} \tag{2.16}$$

Die vorliegende Arbeit orientiert sich an einem Ansatz zur Modellierung von Genregulationsnetzwerken mit kontinuierlichen DBN von Murphy *et al.* [46]. Hier wird vorgeschlagen, mit Hilfe eines Gauß'schen Netzwerks eine Dynamik in Anlehnung an das Additive Regulationsmodell zu modellieren, das regulatorische Interaktionen als unabhängige, additive Ereignisse behandelt. Beide Varianten der Additiven Regulationsmodelle – der lineare und der nichtlineare Ansatz – werden betrachtet. Im Gegensatz zu den Additiven Regulationsmodellen modelliert ein Bayessches Netzwerk die regulatorischen Interaktionen aber als stochastische Beziehungen zwischen den Expressionsraten der Gene und kann so die Tatsache berücksichtigen, daß es aufgrund von Meßfehlern und unbekannten Einflußfaktoren oft nicht möglich ist, deterministische Zusammenhänge zwischen den Expressionsraten der Gene zu identifizieren. So können beide Vorteile – Modellierung der Expressionsraten durch kontinuierliche Variablen und stochastische Beziehungen zwischen den Variablen – in diesem Ansatz vereint werden.
Auch hier arbeitet man vereinfachend mit einem diskreten Zeitsystem und nimmt an, daß sich die Zustände aller Variablen synchron aktualisieren. Analog zu einem diskreten DBN können die dynamischen Prozesse eines Genregulationsnetzwerks als stationäre Markov-Prozesse beschrieben werden, so daß nur ein Zustandsübergang des Netzwerks von einem Zeitpunkt t zu einem Zeitpunkt $t+1$ modelliert werden muß. Die bedingte Wahrscheinlichkeitsdichte einer Variablen $X_i[t+1]$ ergibt sich aus einer Normalverteilung, in deren Mittelwert die auf sie wirkenden regulatorischen Einflüsse additiv eingehen:

linearer Ansatz:

$$X_i[t+1] \sim N(r_i(t) + \beta_i, \sigma^2) = N(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i, \sigma^2) \tag{2.17}$$

oder genauer

$$\begin{aligned} & f(x_i(t+1)|\mathbf{x}(t)) \\ = \; & f(x_i(t+1)|x_1(t), x_2(t), ..., x_N(t)) \\ = \; & \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{\left(x_i(t+1) - (\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i) \right)^2}{2\sigma^2} \right) \end{aligned}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{\left(x_i(t+1) - (\sum_{w_{ij}\neq 0} w_{ij} x_j(t) + \beta_i) \right)^2}{2\sigma^2} \right)$$

$$= f(x_i(t+1)|pa_i) \tag{2.18}$$

nichtlinearer Ansatz:

$$X_i[t+1] \sim N(S(r_i(t)+\beta_i), \sigma^2) = N(S(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i), \sigma^2) \tag{2.19}$$

oder genauer

$$f(x_i(t+1)|\mathbf{x}(t))$$

$$= f(x_i(t+1)|x_1(t), x_2(t), ..., x_N(t))$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{\left(x_i(t+1) - S(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i) \right)^2}{2\sigma^2} \right)$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{\left(x_i(t+1) - \frac{max_i}{1+\exp\left(-\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i\right)} \right)^2}{2\sigma^2} \right)$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(\frac{\left(x_i(t+1) - \frac{max_i}{1+\exp\left(-\sum_{w_{ij}\neq 0} w_{ij} x_j(t) + \beta_i\right)} \right)^2}{2\sigma^2} \right)$$

$$= f(x_i(t+1)|pa_i) \tag{2.20}$$

Analog zu den Additiven Regulationsmodellen geben die genspezifischen Konstanten max_i und b_i die maximale Expressionsrate und den Biasfaktor des Gens g_i an. Die Konstanten w_{ij} beschreiben die Parameter der bedingten Wahrscheinlichkeitsverteilung von x_i. Sie entsprechen den Einträgen der Gewichtsmatrix W im Additiven Regulationsmodell, die die Stärke und die Richtung des jeweiligen regulatorischen

Eigenschaft	dynamisches Verhalten	Variablen	Relationen	Grad der Abstraktion	Zeitsystem
gerichtete Graphen	statisch	–	–	hoch[1]	–
Boolesche Netzwerke	dynamisch	Boolesch	deter-ministisch	hoch	diskret& synchron
diskrete DBN	dynamisch	diskret	stochastisch	hoch	diskret& synchron
Additive Regulations-modelle	dynamisch	kontinu-ierlich	deter-ministisch	hoch	diskret& synchron
kontinuier-liche DBN	dynamisch	kontinu-ierlich	stochastisch	hoch	diskret& synchron

Tabelle 2.2: **Zusammenfassung der genetischen Netzwerkmodelle.** Die Tabelle faßt die betrachteten Netzwerkmodelle bezüglich ihrer Eigenschaften zusammen.

Einflusses der Expression eines Gens g_j auf die Expression eines Gens g_i näher spezifizieren. Die Expression eines Gens g_i wird nicht von der Expression eines Gens g_j beeinflußt, falls der dem jeweiligen Gewicht zugeordnete Wert 0 ist; der entsprechende Summand $w_{ij}x_j(t)$ kann somit aus dem regulatorischen Input $r_i(t)$ gestrichen werden. In der bedingten Wahrscheinlichkeitsdichte $f(x_i(t+1)|pa_i)$ (Gleichung 2.18 bzw. 2.20) wird dann deutlich, daß der Knoten $X_i[t+1]$ nur von seinen Eltern $Pa(X_i[t+1])$ abhängig ist (Bedingte Unabhängigkeit).
Die Integration von extra- und intrazellulären Einflußfaktoren erfolgt analog zu den Additiven Regulationsmodellen.

2.3 Zusammenfassung

In der Tabelle 2.2 sind die in diesem Kapitel vorgestellten Netzwerkmodelle bezüglich ihrer Eigenschaften zusammengefaßt. Für welches Netzwerkmodell man sich im Einzelfall entscheidet, hängt vor allem von der gegebenen Fragestellung und den verfügbaren Expressionsdaten ab: Den einfachsten Ansatz zur Modellierung eines Genregulationsnetzwerks liefern die gerichteten Graphen. Mit ihnen läßt sich allerdings lediglich die Struktur des Genregulationsnetzwerks beschreiben. Möchte man auch Aussagen über sein dynamisches Verhalten treffen, muß ein Netzwerkmodell gewählt werden, das neben der Struktur auch die Dynamik des Genregulationsnetzwerks modelliert. Dabei ist zu entscheiden, ob man die Expressionsraten der Gene durch Boolesche, diskrete oder kontinuierliche Variablen darstellen möchte und ob

[1]Hier bezieht sich der hohe Grad an Abstraktion auf die Tatsache, daß dieses Netzwerkmodell nur allgemein regulatorische Beziehungen zwischen den Genen modelliert, diese aber nicht näher spezifiziert.

die regulatorischen Beziehungen zwischen den Genen durch deterministische oder stochastische Relationen beschrieben werden sollen. Gerade die qualitative Betrachtung der Expressionsraten mit Hilfe von Booleschen Variablen ermöglicht eine einfachere und effizientere Analyse des genetischen Netzwerks, als es die Arbeit mit kontinuierlichen Variablen erlaubt. Bei der dafür erforderlichen Diskretisierung der Expressionsdaten gehen allerdings viele Informationen – vor allem über kleinere Schwankungen der Expressionsraten – verloren. Außerdem ist eine Arbeit mit kontinuierlichen Variablen biologisch realistischer. Die Modellierung der regulatorischen Beziehungen durch stochastische Relationen führt zu komplexeren Modellen als die deterministische Betrachtungsweise, kann wiederum aber Meßfehler und Inkonsistenzen in den Daten besser berücksichtigen.
Aufgrund der Begrenzung derzeit verfügbarer Expressionsdaten auf Messungen der mRNA-Konzentrationen unterliegen alle in diesem Kapitel eingeführten Netzwerkmodelle einem hohen Grad an Abstraktion. Sie beschränken die Genregulationsprozesse auf die Ebene der Transkription, ignorieren alle Regulationsmechanismen auf anderen Ebenen der Genexpression und arbeiten mit der vereinfachten Annahme, daß die Expressionsrate eines Gens allein durch die Konzentration seines mRNA-Transkripts beschrieben werden kann. Möchte man die Prozesse der Genexpression und Genregulation detaillierter modellieren und beispielsweise auch die Protein-Konzentrationen – falls gegeben – berücksichtigen, ist keines der vorgestellten Netzwerkmodelle ohne eine entsprechende Erweiterung geeignet.

Kapitel 3

Reverse Engineering Algorithmen

Nachdem im ersten Schritt des Reverse Engineering Prozesses ein genetisches Netzwerkmodell zur Beschreibung des Genregulationsnetzwerks festgelegt wurde, muß nun für den zweiten Schritt ein geeigneter Reverse Engineering Algorithmus ausgewählt werden. Dieser soll das Netzwerkmodell mit Hilfe der gegebenen Daten „trainieren“ – also die Parameter des Modells geeignet festlegen und spezifizieren, zwischen welchen Netzwerkkomponenten regulatorische Einflüsse bestehen.
Die als Trainingsdaten dienenden Genexpressionsdaten stammen aus den in Abschnitt 1.2 vorgestellten biologischen Experimenten und liefern Informationen über das Expressionsverhalten der Gene in einer Zelle. Wie beschrieben, sind die Protein-Konzentrationen nur schwer und ungenau meßbar, weswegen sich derzeit verfügbare Expressionsdaten oftmals auf Messungen der mRNA-Konzentrationen beschränken. Die vorgestellten Netzwerkmodelle arbeiten deshalb mit vereinfachenden Annahmen, welche die Modellierung der Genexpressionsprozesse auf einer abstrakten Ebene erlauben, in der die Konzentration des zu einem Gen gehörenden mRNA-Transkripts ausreicht, um die Expressionsrate des Gens zu charakterisieren (siehe Abschnitt 2.1). Die Aufgabe eines Reverse Engineering Algorithmus ist es jetzt, Zusammenhänge zwischen den in den Trainingsdaten gegebenen mRNA-Konzentrationen der Gene zu identifizieren und dadurch Rückschlüsse auf regulatorische Interaktionen zwischen ihnen zu ziehen. Auch im folgenden bezieht sich der Begriff „Expressionsrate eines Gens“ daher allein auf die entsprechende mRNA-Konzentration.
Im weiteren Verlauf der Arbeit sollen zur Vereinfachung die Details der einzelnen Technologien zur Generierung der Expressionsdaten – also zur Messung der mRNA-Konzentrationen – nicht weiter betrachtet und stattdessen mit einer allgemeinen Definition eines biologischen Experiments gearbeitet werden. Im besonderen wird dabei vernachlässigt, daß ein Großteil dieser Technologien mit Zellpopulationen und nicht mit einzelnen Zellen arbeitet, und daß es sich bei den gemessenen Werten für die Expressionsraten der Gene nur um die Durchschnittswerte der Zellen in der betrachteten Zellpopulation handelt.

Definition 3.1 (Expressionsexperiment) *Ein Expressionsexperiment dient zur Charakterisierung des genomweiten Genexpressionsmusters einer Zelle zu einem Zeitpunkt t. Es wird also der Systemzustand bestimmt, in dem sich das Genregulationsnetzwerk der Zelle zum Zeitpunkt t befindet. Dafür müssen die Expressionsraten aller interessierenden Gene gleichzeitig gemessen werden.*

Soll das Netzwerkmodell auch extra- und intrazelluläre Einflußfaktoren (Licht, Temperatur, zugesetzte Chemikalien, Hormone, Zytokine, Nährstoffe, etc.) berücksichtigen, so sind in einem Expressionsexperiment außerdem die Ausprägungen der interessierenden Faktoren festzuhalten.
Um den Einfluß einzelner Gene oder Einflußfaktoren auf den Zustand des Genregulationsnetzwerks zu untersuchen, dienen experimentelle Manipulationen:

Definition 3.2 (Manipulationsexperiment) *In einem Manipulationsexperiment wird gezielt das Expressionsverhalten einzelner Gene und/oder der Zustand einzelner Einflußfaktoren mit Hilfe spezieller Verfahren der Gentechnologie*[1] *bzw. durch die Variation der Ausprägung betreffender Einflußfaktoren experimentell manipuliert.*

Die Daten, die bei der Durchführung von Expressionsexperimenten generiert werden, lassen sich im wesentlichen in die drei Gruppen Zustandsübergangsdaten, Zeitreihen und stabile Zustandsdaten einteilen:

Definition 3.3 (Zustandsübergangsdaten) *Hierbei handelt es sich um eine Datenmenge* $D = \{\{\mathbf{x}(t_1), \mathbf{x}(t_1+1)\}, \{\mathbf{x}(t_2), \mathbf{x}(t_2+1)\}, ..., \{\mathbf{x}(t_M), \mathbf{x}(t_M+1))\}\}$*, die M unabhängige Zustandsübergänge des Netzwerks beschreibt. Jedes Paar* $\{\mathbf{x}(t_i), \mathbf{x}(t_i+1)\}$ *besteht dabei aus einem Ausgangszustand* $\mathbf{x}(t_i)$*, der die Expressionsraten der Gene und damit den Systemzustand des Netzwerks zu einem beliebigen Zeitpunkt* t_i *charakterisiert, und seinem zugehörigen Folgezustand* $\mathbf{x}(t_i+1)$*.*

Sind die gemessenen Zustandsübergänge nicht unabhängig voneinander und gilt $\mathbf{x}(t_i+1) = \mathbf{x}(t_{i+1})$, dann entsprechen die Daten einer Zeitreihe:

Definition 3.4 (Zeitreihe) *Bei einer Zeitreihe handelt es sich um eine Datenmenge* $D = \{\mathbf{x}(t_0), \mathbf{x}(t_0+1), ..., \mathbf{x}(t_0+M)\}$ *von zeitlich aufeinanderfolgenden Systemzuständen. Ausgehend von einem Anfangszustand* $\mathbf{x}(t_0)$ *zu einem beliebigen Zeitpunkt* t_0 *werden mit Hilfe von Expressionsexperimenten die in einem Zeitabstand von* Δt *aufeinanderfolgenden Systemzustände gemessen und protokolliert.*

Idealisierter Weise sollte die zwischen dem Ausgangszustand $\mathbf{x}(t_i)$ und dem Folgezustand $\mathbf{x}(t_i+1)$ eines Zustandsübergangspaares bzw. zwischen zwei aufeinanderfol-

[1] Mit Hilfe experimenteller Verfahren kann die Expression eines Gens gezielt manipuliert werden. Es existieren Methoden zur Überexpression, aber auch zur Hemmung der Expression spezifischer Gene [37].

genden Systemzuständen einer Zeitreihe liegende Zeitspanne Δt genau so lang sein, daß jedes Gen des Netzwerks seine Expressionsrate exakt einmal aktualisieren kann. Die Tatsache, daß einem Genregulationsnetzwerk ein asynchrones, kontinuierliches Zeitsystem zugrunde liegt, macht es in der Praxis allerdings schwer, einen geeigneten Wert für Δt festzulegen. Wählt man Δt zu klein, reicht die Zeitspanne für einige Gene vielleicht nicht aus, um ihre Expressionsrate zu aktualisieren. Ist Δt hingegen zu groß, können sich viele Gene in diesem Zeitraum sogar mehrfach aktualisieren. Bei der Wahl eines geeigneten Δt sind außerdem die Zerfallsraten der Genprodukte zu berücksichtigen. Ist Δt zu groß, ist ein erheblicher Teil bestimmter Genprodukte in diesem Zeitraum bereits wieder zerfallen und kann nicht mehr nachgewiesen werden.
Um diese Schwierigkeiten zu umgehen, konzentriert man sich in manchen Fällen mehr auf das Langzeitverhalten eines Genregulationsnetzwerks und arbeitet mit sogenannten stabilen Zustandsdaten (*engl.: steady state data*). Zur Generierung dieser Daten werden Expressions- und Manipulationsexperimente (Definitionen 3.1 und 3.2) miteinander kombiniert. Man geht davon aus, daß sich das Netzwerk einer differenzierten Zelle in einem stabilen Attraktor befindet und auch nach der experimentellen Manipulation einzelner Komponenten wieder in einen solchen hinein läuft. Ein Vergleich der beiden Attraktorzustände soll zur Identifizierung der Gene dienen, die aufgrund der Manipulation ihr Expressionsverhalten verändert haben. Probleme entstehen dabei aufgrund der Tatsache, daß Genregulationsnetzwerke Zyklen enthalten können und es sich bei den Attraktorzuständen eines zyklischen Netzes oft um stabile Zyklen und nicht um stationäre Zustände handelt. Der Vergleich der beiden Attraktorzustände wird dann erschwert, denn Änderungen im Expressionsverhalten eines Gens können auch durch die zyklische Änderung seiner Expressionsrate im stabilen Zyklus hervorgerufen worden sein und müssen nicht aus der experimentellen Manipulation resultieren. In der Praxis werden diese Probleme dadurch gelöst, daß man neben der zu manipulierenden Zelle auch eine Zelle („Kontrollzelle“) betrachtet, in der keine Manipulation erfolgt. Zu dem Zeitpunkt t_0 unmittelbar vor der Manipulation befindet sich das Genregulationsnetzwerk in der zu manipulierenden Zelle im gleichen Zustand, wie das Genregulationsnetzwerk der Kontrollzelle. Nach einem genügend großem Zeitraum ΔT, in dem die manipulierte Zelle wieder in einen Attraktor gelangen konnte, bestimmt man mit Hilfe eines Expressionsexperiments sowohl das Expressionsmuster $\mathbf{x}(t_0 + \Delta T)$ der manipulierten Zellen als auch das Expressionsmuster $\mathbf{k}(t_0 + \Delta T)$ der Kontrollzelle und kann über den Vergleich dieser beiden Expressionsmuster genau die Gene identifizieren, die ihr Expressionsverhalten aufgrund der Manipulation verändert haben:

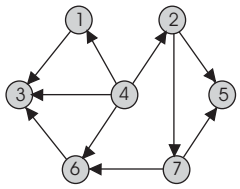
Definition 3.5 (stabile Zustandsdaten) *Hierbei handelt es sich um eine Datenmenge* $D = \{\{\mathbf{x}_1(t_0 + \Delta T), \mathbf{k}_1(t_0 + \Delta T)\}, \{\mathbf{x}_2(t_0 + \Delta T), \mathbf{k}_2(t_0 + \Delta T)\}, ..., \{\mathbf{x}_M(t_0 + \Delta T), \mathbf{k}_M(t_0 + \Delta T)\})\}$*, die die Ergebnisse aus* M *verschiedenen Manipulationsexperimenten präsentiert. Ein Paar* $\{\mathbf{x}_i(t_0 + \Delta T), \mathbf{k}_i(t_0 + \Delta T)\}$ *enthält zum einen den*

stabilen Systemzustand $\mathbf{x}_i(t_0 + \Delta T)$ *zum Zeitpunkt* $t_0 + \Delta T$*, in den das Genregulationsnetzwerk der Zelle nach einem bestimmten Manipulationsexperiment, das zum Zeitpunkt* t_0 *stattfand, gelangt ist, und den stabilen Systemzustand* $\mathbf{k}_i(t_0 + \Delta T)$ *des Genregulationsnetzwerks einer Kontrollzelle zum selben Zeitpunkt.*

In diesem Kapitel soll darauf eingegangen werden, wie mit Hilfe eines Reverse Engineering Algorithmus die in Abschnitt 2.2 vorgestellten Netzwerkmodelle an solche Expressionsdaten angepaßt werden können. Für jedes Netzwerkmodell sind zunächst die speziellen Anforderungen an einen Reverse Engineering Algorithmus einleitend dargestellt. Anschließend wird dann ein ausgewählter Reverse Engineering Algorithmus für dieses Netzwerkmodell detailliert betrachtet und analysiert, welche Limitationen sich für diesen Algorithmus – neben der vereinfachten Betrachtungsweise der Genregulationsprozesse – aufgrund bestimmter Eigenschaften des Netzwerkmodells (modellbedingte Limitationen) oder bestimmter Eigenschaften des Algorithmus selbst (algorithmusbedingte Limitationen) ergeben.
Da die Modellierung des regulatorischen Einflusses eines extra- bzw. intrazellulären Faktors auf die Expression eines Gens in einem abstrakten Netzwerkmodell identisch zu der Modellierung eines regulatorischen Einflusses erfolgt, den die Expression eines Gens auf die Expression eines anderen Gens ausübt, muß die Rekonstruktion der regulatorischen Einflüsse solcher Faktoren hier nicht gesondert betrachtet werden. Im folgenden werden diese Faktoren daher vernachlässigt.

3.1 Reverse Engineering in Gerichteten Graphen

Ein gerichteter Graph G modelliert ein Genregulationsnetzwerk durch das Paar $\langle V, E\rangle$. Die Menge $V = \{1, 2, ..., N\}$ der Knoten entspricht dabei der Menge $\{g_1, g_2, ..., g_N\}$ aller Gene des Genregulationsnetzwerks; die Menge E aller gerichteten Kanten zwischen den Knoten den direkten regulatorischen Interaktionen zwischen den Genen. Eine zu dem Paar $\langle V, E\rangle$ äquivalente Darstellung des gerichteten Graphen G liefert die Adjazenzliste Adj_G (Abbildung 3.1(b)). Für jeden Knoten i enthält das Listenelement $Adj_G(i)$ die Menge aller Knoten von G, die adjazent zu Knoten i sind. Ein Knoten j ist adjazent zu Knoten i, wenn eine gerichtete Kante $i \rightarrow j$ von i nach j existiert.
Einem Reverse Engineering Algorithmus zur Rekonstruktion eines Genregulationsnetzwerks stellt sich die Aufgabe, regulatorische Interaktionen zwischen den Genen aufgrund von Zusammenhängen zwischen ihren Expressionsraten zu identifizieren. Übertragen auf den gerichteten Graphen als Modell für das Genregulationsnetzwerk bedeutet dies: Die Menge V der Knoten ist bekannt, und es gilt, die Menge E der Kanten zwischen den Knoten festzulegen. Für jeden Knoten i von G ist also das entsprechende Listenelement $Adj_G(i)$ der Adjazenzliste Adj_G von G zu spezifizieren. Reverse Engineering Algorithmen arbeiten hier in der Regel mit stabilen Zustandsdaten (siehe Definition 3.5)



(a)

Adjazenzliste Adj

1: 3
2: 5,7
3:
4: 1,2,3,6
5:
6: 3
7: 5,6

(b)

Erreichbarkeitsliste Acc

1: 3
2: 3,5,6,7
3:
4: 1,2,3,5,6,7
5:
6: 3
7: 3,5,6

(c)

Abbildung 3.1: **Adjazenz- und Erreichbarkeitsliste eines gerichteten Graphen:** (a) graphische Darstellung, (b) Adjazenzliste, (c) Erreichbarkeitsliste.

$$
\begin{aligned}
D &= \{\{\mathbf{x}_1, \mathbf{k}_1\}, ..., \{\mathbf{x}_N, \mathbf{k}_N\}\} \\
&= \left\{ \left\{ \begin{pmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_N^1 \end{pmatrix}, \begin{pmatrix} k_1^1 \\ k_2^1 \\ \vdots \\ k_N^1 \end{pmatrix} \right\}, \cdots, \left\{ \begin{pmatrix} x_1^N \\ x_2^N \\ \vdots \\ x_N^N \end{pmatrix}, \begin{pmatrix} k_1^N \\ k_2^N \\ \vdots \\ k_N^N \end{pmatrix} \right\} \right\},
\end{aligned}
$$

die die Ergebnisse aus N Manipulationsexperimenten beschreiben. Sie dienen zur Generierung der sogenannten Erreichbarkeitsliste (*engl.: accessibility list*) Acc_G von G (Abbildung 3.1(c)), deren Listenelemente $Acc_G(i)$ jeweils die Gene enthalten, die ihr Expressionsverhalten nach der Manipulation des entsprechenden Gens g_i verändert haben. In jedem Manipulationsexperiment wurde genau eines der N Gene des Netzwerks gestört. Zur Konstruktion von $Acc_G(i)$ muß der Attraktorzustand $\mathbf{x}_i$, in den die Zelle nach der Manipulation der Expression von Gens g_i gelangte, mit dem Attraktorzustand $\mathbf{k}_i$ einer Kontrollzelle, die keiner Manipulation unterlag, verglichen werden. Im Gegensatz zu der Adjazenzliste enthält ein Listenelement $Acc_G(i)$ der Erreichbarkeitsliste neben den Genen, deren Expression direkt von der Expression des Gens g_i beeinflußt wird, auch die Gene, deren Expression nur über indirekte Interaktionen von der Expression des Gens g_i abhängt. Als Beispiel stelle man sich folgende Situation vor: Angenommen, das Genprodukt von Gen g_i bindet an eine entsprechende Kontrollregion von Gen g_j und bewirkt so eine erhöhte Expression von g_j. Das Produkt von Gen g_j wiederum bewirkt die Phosphorilierung eines anderen Genproduktes, das dadurch in einen aktiven Zustand überführt wird, sich an eine Kontrollregion von Gen g_k binden kann und so die Expression von g_k unterdrückt. Beide Gene g_j und g_k werden ihr Expressionsverhalten nach einer Manipulation von Gen g_i ändern und so in dem Listenelement $Acc_G(i)$ enthalten sein, auch wenn nur die Expression von Gen g_j direkt durch die Expression von Gen g_i reguliert und die Expression von Gen g_k lediglich indirekt beeinflußt wird. Aus Sicht der Graphentheorie gibt die Erreichbarkeitsliste für jeden Knoten i die Menge der Knoten an, die über eine Folge von Kanten (Pfad) von i aus erreichbar sind. Sie entspricht

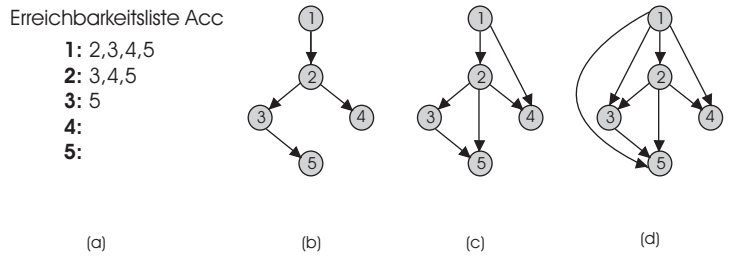


Abbildung 3.2: **Graphen mit gleicher Erreichbarkeitsliste:** (a): Erreichbarkeitsliste, (b): minimaler Graph G_{min} (c),(d): weitere zu *Acc* kompatible Graphen.

annähernd der transitiven Hülle von G mit dem Unterschied, daß das Listenelement $Acc_G(i)$ den Knoten i nie selbst enthält[2].
Die Aufgabe eines Reverse Engineering Algorithmus ist es, aus dieser Menge der in den Trainingsdaten beobachteten direkten und indirekten Interaktionen die direkten Interaktionen zu selektieren und so die Adjazenzliste des Graphen zu konstruieren. Dabei taucht folgendes Problem auf: Die Erreichbarkeitsliste eines Graphen G ist zwar eindeutig durch seine Adjazenzliste definiert. Die Umkehrung dieser Aussage gilt jedoch nicht – in der Regel gibt es mehrere Graphen, die die gleiche Erreichbarkeitsliste besitzen (Abbildung 3.2). Allerdings definiert jede Erreichbarkeitsliste *Acc* genau einen Graphen, der kompatibel mit *Acc* ist (d.h.: seine Erreichbarkeitsliste Acc_G ist äquivalent zu Acc) und der weniger Kanten besitzt als alle anderen, zu Acc kompatiblen Graphen. Dieser Graph wird auch als minimaler Graph G_{min} (*engl.: most parsimonious graph*) bezeichnet [63] (Abbildung 3.2 (b)). Im Unterschied zu allen anderen Graphen, die mit einer gegebenen Erreichbarkeitsliste *Acc* kompatibel sind, enthält der minimale Graph G_{min} keine sogenannten Kurzwege (*engl.: shortcuts*), die wie folgt definiert sind:

Definition 3.6 (Spannweite r einer Kante (*engl.: range r*) [63]) *Sei e eine Kante zwischen zwei Knoten i und j. Die Spannweite r der Kante e ist die Länge des kürzesten Weges zwischen i und j, der nicht über e führt. Gibt es keinen anderen Weg von i nach j, dann ist die Spannweite r der Kante e unendlich ($r = \infty$).*

Definition 3.7 (Kurzweg [63]) *Eine Kante e mit einer Spannweite $r \geq 2$ und $r \neq \infty$ wird auch als Kurzweg bezeichnet.*

Es soll nun ein von Wagner [63] entworfener Reverse Engineering Algorithmus für die Lösung dieser Problematik vorgestellt werden. Ein ähnlicher Ansatz ist in [47] nachzulesen.

[2]Ein Gen g_i kann sich zwar schon selbst regulieren (Autoregulation), dieser Einfluß ist hier aber nicht nachweisbar: Nach der Manipulation eines Gens g_i, verbleibt dieses in dem erzwungenen Zustand und kann so selbst nicht auf die Manipulation reagieren.

3.1.1 Adjazenzlisten - Konstruktion (Wagner [63])

Ausgehend von gegebenen stabilen Zustandsdaten generiert dieser Algorithmus zunächst die Erreichbarkeitsliste Acc_G und konstruiert anschließend durch das Entfernen aller Kurzwege die Adjazenzliste $Adj_{G_{min}}$ des durch Acc_G definierten minimalen Graphen G_{min}.

Der Algorithmus

Als erste Teilaufgabe ist also die entsprechende Erreichbarkeitsliste Acc_G zu generieren. Hierzu werden in [63] keine Angaben gemacht. Um das Listenelement $Acc_G(i)$ für einen Knoten i zu erstellen, muß der Algorithmus prinzipiell für jedes Gen $g_j, j \neq i$ entscheiden, ob es sein Expressionsverhalten nach der Manipulation von Gen g_i verändert hat. Dafür wird die Expressionsrate k_j^i von g_j im Attraktorzustand $\mathbf{k}_i$ der nicht manipulierten Kontrollzelle mit der Expressionsrate x_j^i von g_j im Attraktorzustand $\mathbf{x}_i$ der manipulierten Zelle verglichen. Da die Daten aus biologischen Experimenten durch gewisse Meßfehler verrauscht sind, kann ein Unterschied zwischen k_j^i und x_j^i nicht unbedingt als Beweis dafür gelten, daß Gen g_j sein Expressionsverhalten verändert hat. Oftmals liegen mehrere Messungen der Attraktorzustände $\mathbf{k}_i$ und $\mathbf{x}_i$ vor. Dann kann ein statistischer Test herangezogen werden, um zu entscheiden, ob ein beobachteter Unterschied zwischen den empirischen Mittelwerten $\bar{k}_j^i$ und $\bar{x}_j^i$ nur auf zufällige Schwankungen oder tatsächlich auf eine Änderung der Expressionsrate von g_j zurückzuführen ist. Dazu testet man die Nullhypothese, daß die gemittelte Expressionsrate $\bar{k}_j^i$ gleich der gemittelten Expressionsrate $\bar{x}_j^i$ ist:

$$H_0 : \qquad \bar{k}_j^i = \bar{x}_j^i \tag{3.1}$$

Unter der Annahme, daß der Fehler in den Daten normalverteilt ist, kann mit einem t-Test gearbeitet werden. Die Nullhypothese wird abgelehnt, falls:

$$T(\bar{k}_j^i, \bar{x}_j^i) = \frac{|\bar{k}_j^i - \bar{x}_j^i|}{\sqrt{\left[\frac{n_k+n_x}{n_k \cdot n_x}\right] \cdot \left[\frac{(n_k-1)s_k^2+(n_x-1)s_x^2}{n_k+n_x-2}\right]}} > t^{zweis}_{\alpha, n_k+n_x-2} \tag{3.2}$$

Hierbei geben n_k und n_x die Anzahl der Messungen von k_j^i und x_j^i an; s_k und s_x beschreiben ihre Standardabweichungen. Die Werte für $t^{zweis}_{\alpha, n_k+n_x-2}$ können aus einer entsprechenden Tabelle entnommen werden[3]. Eine Ablehnung der Nullhypothese bedeutet, daß sich der beobachtete Unterschied zwischen $\bar{k}_j^i$ und $\bar{x}_j^i$ nicht nur auf zufällige Schwankungen zurückführen läßt. Man kann dann annehmen, daß Gen g_j sein Expressionsverhalten nach der Störung von Gen g_i tatsächlich verändert hat und darf so den Knoten j in das Listenelement $Acc_G(i)$ eingetragen. Wichtig ist hier eine geeignete Wahl des Signifikanzniveaus α. Mit α wird die Wahrscheinlichkeit angegeben, beim Testen einen Fehler zu begehen und die richtige Nullhypothese

[3]Eine solche Tabelle ist beispielsweise in [52] zu finden.

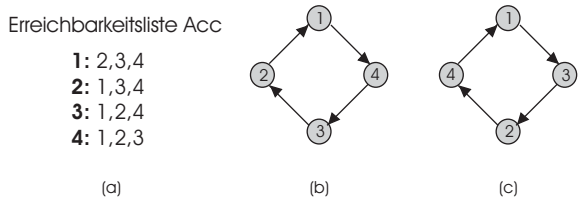


Abbildung 3.3: **Erreichbarkeitsliste eines Zyklus:** Jede Reihenfolge der Knoten in einem Zyklus erzeugt die gleiche Erreichbarkeitsliste. Damit ist die Reihenfolge der Konten nicht eindeutig aus einer gegebenen Erreichbarkeitsliste rekonstruierbar.

abzulehnen (Fehler 1. Art). Der Knoten j wird dann in das Listenelement $Acc_G(i)$ aufgenommen, obwohl Gen g_j sein Expressionsverhalten nicht verändert hat und der Unterschied zwischen $\bar{k}_j^i$ und $\bar{x}_j^i$ nur zufälliger Natur ist. Demgegenüber steht die Wahrscheinlichkeit β, einen Fehler 2. Art zu begehen und die falsche Nullhypothese beizubehalten. Der Knoten j wird dann nicht in das Listenelement $Acc_G(i)$ eingetragen, obwohl Gen g_j sein Expressionsverhalten verändert hat. Beide Fehlerwahrscheinlichkeiten α und β sind sowohl voneinander als auch von der Anzahl der verfügbaren Meßwerte n_k und n_x abhängig.
Aufgrund von Fehlern, die beim Testen begangen werden, kann es passieren, daß die entstehende Erreichbarkeitsliste Acc_G nicht konsistent ist. Eine konsistente Erreichbarkeitsliste Acc_G zeichnet sich durch die Eigenschaft der Transitivität aus: Ist Knoten j in der Menge $Acc_G(i)$ enthalten und Knoten k in der Menge $Acc_G(j)$, dann muß Knoten k ebenfalls in $Acc_G(i)$ zu finden sein, falls $k \neq i$. In Schritt zwei muß der Algorithmus die im ersten Schritt generierte Erreichbarkeitsliste Acc_G bezüglich dieser Eigenschaft überprüfen und gegebenenfalls fehlende Einträge ergänzen, da sonst im weiteren Verlauf Komplikationen auftreten können.
Probleme ergeben sich auch, wenn das Netzwerk nicht azyklisch ist. In einem Genregulationsnetzwerk können zwei Arten von Zyklen auftreten. Zum einen gibt es Autoregulationen, d.h. bestimmte Gene können ihre Expression selbst direkt regulieren. Solche Autoregulationen müssen hier nicht betrachtet werden, denn sie lassen sich mit einem Manipulationsexperiment nur schwer identifizieren: Nach der Manipulation eines Gens g_i verbleibt dieses in dem erzwungenen Zustand und kann so selbst nicht auf die Manipulation reagieren [63]. Zum anderen lassen sich auch Zyklen finden, in denen mehrere Gene involviert sind. Diese Art von Zyklen werden auch als Rückkopplungsschleifen (*engl.: feedback loops*) bezeichnet und stellen eine Möglichkeit dar, mit der Gene über indirektem Wege regulierend auf ihre eigene Expression einwirken können. Abbildung 3.3 verdeutlicht das Problem, das beim Auftreten dieser Art von Zyklus entsteht: Jede Reihenfolge der in einem Zyklus involvierten Gene liefert dieselbe Erreichbarkeitsliste *Acc*, denn die Störung eines jeden Gens im Zyklus hat Auswirkungen auf alle anderen Gene des Zyklus. Damit ist die Struktur eines Zyklus aus der Erreichbarkeitsliste *Acc* allein nicht eindeutig

rekonstruierbar. Bevor der Algorithmus im eigentlichem Hauptteil die Adjazenzliste konstruieren kann, muß er also zuerst einen azyklischen Graphen G_{azyk} generieren, indem er alle Zyklen des Netzwerks identifiziert und die jeweiligen Knoten eines Zyklus zu einer Komponente zusammenfaßt. Er verwendet dazu den folgenden Satz:

Satz 3.1 ([63]) *Seien i und j ($i \neq j$) zwei Knoten eines gerichteten Graphen G. i und j gehören zu einem Zyklus, wenn $i \in Acc(j)$ und $j \in Acc(i)$.*

Die Knoten des Graphen G_{azyk} entsprechen damit Mengen von Knoten des ursprünglichen Graphen G (also Mengen von Genen): Ein Knoten i, der in keinem Zyklus involviert ist, wird auf die einelementige Menge $\{i\}$ abgebildet; alle Knoten eines Zyklus werden zu einer mehrelementigen Menge zusammengefaßt.
Im eigentlichen Hauptteil des Algorithmus gilt es nun, die Adjazenzliste $Adj_{G_{azyk}}$ des gerichteten Graphen G_{azyk} aus der erstellten Erreichbarkeitsliste $Acc_{G_{azyk}}$ so zu konstruieren, daß G_{azyk} keine Kurzwege enthält und damit ein minimaler Graph ist. Diese Konstruktion beruht auf folgendem Satz:

Satz 3.2 ([63]) *Seien i, j und k drei paarweise verschiedene Knoten eines azyklischen, frei von Kurzwegen, gerichteten Graphen G. Ist j erreichbar von i, dann gibt es keinen Knoten k, der von j erreichbar und adjazent zu i ist.*

Ausgehend von der Erreichbarkeitsliste werden rekursiv alle Einträge entfernt, die zu diesem Satz inkonsistent sind.

Zusammenfassend sind hier noch einmal die Teilschritte des Algorithmus aufgelistet:

1. Generiere aus den gegeben stabilen Zustandsdaten mit Hilfe des t-Tests die Erreichbarkeitsliste Acc_G.
2. Stelle die Transitivität von Acc_G sicher.
3. Eliminiere alle Zyklen: $Acc_G \Longrightarrow Acc_{G_{azyk}}$.
4. Konstruiere aus der Erreichbarkeitsliste $Acc_{G_{azyk}}$ die Adjazenzliste $Adj_{G_{azyk}}$ von G_{azyk} so, daß dieser keine Kurzwege enthält und ein minimaler Graph ist.

Implementierung

Für den Algorithmus kann folgender Pseudocode angegeben werden:

```
1     FOR EACH node i of G
2         Acc_G(i) = ∅
```

```
3     FOR EACH node i of G
4         FOR EACH node j of G
5             IF i ≠ j
6                 IF T(x̄_j^0, x̄_j^i) > t^{zweis}_{α,n0+ni−2}
7                     add j to Acc_G(i)

8     FOR EACH node i of G
9         FOR EACH node j of G
10            IF i ∈ Acc_G(j)
11                FOR EACH node k of G
12                    IF k ∈ Acc_G(i)
13                        IF j ≠ k
14                            IF k ∉ Acc_G(j)
15                                add k to Acc_G(j)

16    FOR EACH node i of G
17        IF component[i] has not been defined
18            create a new node x of G_azyk
19            component[i] = x
20            FOR EACH node j in Acc_G(i)
21                IF i ∈ Acc_G(j)
22                    component[j] = x

23    FOR EACH node i of G_azyk
24        Acc_{G_azyk}(i) := ∅
25    FOR EACH node i of G
26        FOR EACH node j in Acc_G(i)
27            IF component[i] ≠ component[j]
28                IF component[j] ∉ Acc_{G_azyk}(component[i])
29                    add component[j] to Acc_{G_azyk}(component[i])

30    FOR EACH node i of G_azyk
31        Adj_{G_azyk}(i) = Acc_{G_azyk}(i)
32    FOR EACH node i of G_azyk
33        IF node i has not been checked
34            call CHECK_ACC(i)

35    PROCEDURE CHECK_ACC(i)
36        FOR EACH node j ∈ Acc_{G_azyk}(i)
37            IF Acc_{G_azyk}(j) = ∅
38                declare j as checked
39            ELSE
40                call CHECK_ACC(j)

41        FOR EACH node j ∈ Acc_{G_azyk}(i)
42            FOR EACH node k ∈ Adj_{G_azyk}(j)
43                IF k ∈ Acc_{G_azyk}(i)
44                    delete k from Adj_{G_azyk}(i)
45        declare node i as checked
```

Der erste Teil des Algorithmus (Zeilen 1-7) dient zur Generierung der Erreichbarkeitsliste Acc_G aus den stabilen Zustandsdaten.
Wie beschrieben ähnelt die Erreichbarkeitsliste Acc_G der transitiven Hülle von G. Um die generierte Erreichbarkeitsliste Acc_G im zweiten Teil (Zeilen 8-15) auf Konsistenz zu überprüfen und fehlende Einträge gegebenenfalls einzufügen, kann deshalb der Algorithmus von Warshall [22] zur Erzeugung der transitiven Hülle eines Graphen verwendet werden.
Die nächsten beiden Teile des Algorithmus dienen zur Generierung des azyklischen Graphen G_{azyk}: Der dritte Teil (Zeilen 16-22) erzeugt zunächst die Knoten von G_{azyk}, indem er jedem Knoten in G einen Knoten in G_{azyk} zuordnet. Die Zuordnung erfolgt über das Feld *component*. Zu Beginn ist der Feldeintrag *component[i]* für jeden Knoten i in G undefiniert; am Ende des dritten Teils gibt *component[i]* an, auf welchen Knoten in G_{azyk} der Knoten i abgebildet wurde. Knoten in G, die zu einem Zyklus gehören, werden unter Anwendung des Satzes 3.1 in G_{azyk} zu einer Komponente zusammengefaßt (Zeilen 20-22). Der vierte Teil (Zeilen 23-29) erzeugt die Erreichbarkeitsliste $Acc_{G_{azyk}}$ für den Graphen G_{azyk}.
Schließlich konstruiert der Algorithmus im fünften Teil (Zeilen 30-34) mit Hilfe der Prozedur *CHECK_ACC* (Zeilen 35-45) aus der Erreichbarkeitsliste $Acc_{G_{azyk}}$ die Adjazenzliste $Adj_{G_{azyk}}$ des Graphen G_{azyk}. Jedes Adjazenzlistenelement $Adj_{G_{azyk}}(i)$ wird zuerst mit dem zugehörigen Erreichbarkeitslistenelement $Acc_{G_{azyk}}(i)$ initialisiert und damit die Beziehung $Adj(i) \subseteq Acc(i)$ ausgenutzt. Anschließend verwendet die Prozedur *CHECK_ACC* den Satz 3.2, um rekursiv Einträge in $Adj_{G_{azyk}}$ zu entfernen (Zeilen 41-44) und die Adjazenzliste $Adj_{G_{azyk}}$ schrittweise zu der eines minimalen, kurzwegfreien Graphen zu verkleinern.
Eine genaue Erklärung zu den Teilen $3-5$ ist in [63] nachzulesen.

Limitationen

Abschließend soll untersucht werden, welche Faktoren die Güte der von diesem Reverse Engineering Algorithmus erzeugten Ergebnisse beeinflussen.
Eine wichtige Rolle spielen hier die Zyklen des Genregulationsnetzwerks. Zum einen ist es nicht möglich, die Struktur eines Zyklus zu rekonstruieren. Ferner sind auch identifizierte Kanten von oder zu einem Zyklus nicht sehr nützlich. Dies gilt besonders dann, wenn viele Gene in einem Zyklus involviert sind, da es nicht möglich ist zu bestimmen, von welchem Gen im Zyklus der regulatorische Einfluß ausgeht bzw. welches Gen im Zyklus von ihm beeinflußt wird. Um diese Aussagen treffen zu können, sind zusätzliche biologische Experimente erforderlich.
Eine weitere wichtige Frage betrifft den Sachverhalt, wie korrekt die Erreichbarkeitsliste erstellt werden kann. Dies hängt vor allem davon ab, wie fehlerbehaftet die Trainingsdaten sind. Bei der Generierung der Erreichbarkeitsliste ist ein statistischer Test erforderlich, der über die Einträge in die Erreichbarkeitsliste entscheidet. Dabei geht man notwendigerweise das Risiko ein, eine Fehlentscheidung zu treffen und

fälschlicherweise ein Gen g_j in das Erreichbarkeitslistenelement $Acc_G(i)$ aufzunehmen, obwohl g_j nicht von Gen g_i abhängt (Fehler 1. Art). Auch kann es passieren, daß man fälschlicherweise ein Gen g_j nicht in das Erreichbarkeitslistenelement $Acc_G(i)$ aufnimmt, obwohl g_j von Gen g_i beeinflußt wird (Fehler 2. Art). Je fehlerbehafteter die Trainingsdaten sind, desto größer wird bei gleicher Irrtumswahrscheinlichkeit α für einen Fehler 1. Art die Irrtumswahrscheinlichkeit β für einen Fehler 2. Art. An dieser Stelle sei drauf hingewiesen, daß fehlende Elemente in Acc_G (Fehler 2. Art) weniger problematisch sind als zusätzliche Elemente (Fehler 1. Art), denn durch die Absicherung der Transitivität können fehlende Elemente auch nachträglich in die Erreichbarkeitsliste eingefügt werden. Dagegen führen zusätzliche Elemente in Acc_G bei der Überprüfung der Transitivität möglicherweise zu weiteren falschen Einträgen. Deshalb wurde bei den Simulationsexperimenten in dieser Arbeit mit einer sehr kleinen Irrtumswahrscheinlichkeit für einen Fehler 1. Art gearbeitet: $\alpha = 0.001$. Weiterhin benutzt der Algorithmus die Annahme, daß die Struktur eines Genregulationsnetzwerks minimal ist und generiert die Adjazenzliste des entsprechenden Graphen so, daß dieser frei von Kurzwegen ist. Es bleibt zu untersuchen, inwiefern diese Annahme der biologischen Realität entspricht.

3.2 Reverse Engineering in Booleschen Netzwerken

Ein Boolesches Netzwerk modelliert die Expressionsraten der Gene idealisiert durch Boolesche Variablen. Gegebene Expressionsdaten aus biologischen Experimenten müssen hier deshalb zuerst entsprechend diskretisiert werden. Die maximale Expressionsrate max_i eines Gens g_i bekommt dazu die höchste in den gegebenen Daten beobachtete Expressionsrate eines Gens g_i zugewiesen. Auf Basis des Schwellwertes $\frac{max_i}{2}$ kann jede gemessene Expressionsrate von g_i auf die Zustände „0“ und „1“ der Booleschen Variablen x_i abgebildet werden.
Die Aufgabe eines Reverse Engineering Algorithmus besteht darin, mit Hilfe dieser diskretisierten Daten für jede Boolesche Variable x_i eine Elternmenge $Pa(x_i) = \{x_{i1}, x_{i2}, ..., x_{ik}\}$ von Booleschen Variablen zu selektieren, deren Zustände zu einem beliebigen Zeitpunkt t den Zustand von x_i zum Folgezeitpunkt $t+1$ bedingen. Anschließend muß dann eine entsprechende Boolesche Funktion f_i spezifiziert werden, die die Abhängigkeit zwischen x_i und seiner Elternmenge beschreibt. Die Booleschen Variablen $x_{i1}, x_{i2}, ..., x_{ik}$ bezeichnet man auch als Inputelemente der Booleschen Funktion f_i und ihre Zustände als Inputzustände. Analog dazu heißt die Variable x_i auch Outputelement von f_i und ihr Zustand Outputzustand.
Boolesche Netzwerke gehörten zu den ersten genetischen Netzwerkmodellen, für die entsprechende Reverse Engineering Algorithmen existierten. Bekannte Algorithmen stammen von Akutsu *et al.* [2, 3, 4] sowie Ideker *et al.* [29]. Diese versuchen, die regulatorischen Interaktionen mit Hilfe von stabilen Zustandsdaten aus verschie-

denen Manipulationsexperimenten zu erlernen, bei denen jeweils ein oder mehrere Gene gleichzeitig überexprimiert oder gehemmt werden. Für jedes Gen g_i läßt sich eine minimale Menge anderer Gene bestimmen, die die in den Daten beobachteten Unterschiede im Expressionsverhalten von g_i erklärt und so als seine Elternmenge dient.
Der wohl wichtigste Reverse Engineering Algorithmus in dem Bereich der Booleschen Netzwerke ist der *Reveal* Algorithmus. Er soll hier näher vorgestellt werden.

3.2.1 Reveal (Liang et al. [36])

Der Reverse Engineering Algorithmus *Reveal* benutzt diskretisierte Zustandsübergangsdaten, um mit Hilfe von Techniken aus der Informationstheorie regulatorische Interaktionen zwischen den Genen zu identifizieren. Gegeben sind also Paare von Systemzuständen, die jeweils aus einem Anfangszustand und dem Folgezustand des Genregulationsnetzwerks bestehen (vergleiche Definition 3.3). Ein Anfangszustand beschreibt die Expressionsraten der Gene zu einem bestimmten Zeitpunkt t und kann auch als Inputzustand des Systems bezeichnet werden; der entsprechende Folgezustand enthält dann die aktualisierten Expressionsraten der Gene zum Zeitpunkt $t + 1$ und heißt auch Outputzustand des Systems.
Für eine Boolesche Variable x_i werden alle möglichen Kombinationen von k Inputelementen ($1 \leq k \leq N$) getestet, bis eine von ihnen x_i als Outputelement einer Booleschen Funktion vollständig bestimmt und somit die Elternmenge $Pa(x_i)$ bildet. Die Boolesche Funktion f_i spezifiziert man anschließend durch das Eintragen der in den Trainingsdaten beobachteten Zusammenhänge zwischen der Elternmenge $Pa(x_i)$ von Inputelementen und dem Outputelement x_i in eine Regeltabelle.

Informationstheoretische Grundlagen

Die Informationstheorie beschäftigt sich mit den mathematischen und statistischen Grundlagen der Nachrichtenübertragung. Eine der Hauptfragestellungen der Informationstheorie betrifft die Definition und Quantifizierung von Information und Unsicherheit. Als ihr wesentlicher Begründer gilt Shannon [55].
Die Datenübertragung von einem Sender über einen störungsbehafteten Kanal zu einem Empfänger stellt die zentrale Problemstellung der klassischen Informationstheorie dar. Der Sender selbst wird dabei als Informations- oder Nachrichtenquelle bezeichnet, die mit einer bestimmten Wahrscheinlichkeitsverteilung P nacheinander einzelne Symbolen aus einer n-elementigen Menge $\{x_1, x_2, ..., x_n\}$ erzeugt. Ein Symbol x_i kann somit als Realisierung einer Zufallsvariablen X angesehen werden.
Die Entropie einer Quelle bzw. der von ihr realisierten Zufallsvariablen X gibt den durchschnittlichen Informationsgewinn an, der sich aus der Beobachtung eines von der Quelle generierten Symbols ziehen läßt. Der Informationsgewinn ist äquivalent zu der Unsicherheit über das Symbol, das die Quelle als nächstes erzeugen wird.

Die Entropie als ein zentrales Maß der Informationstheorie ist durch die folgende Gleichung definiert:

Definition 3.8 (Entropie einer Zufallsvariablen)

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

p_i beschreibt hier die Wahrscheinlichkeit, daß die Quelle das Symbol x_i erzeugt.
Die Unsicherheit über das nächste Symbol ist am größten, wenn die Quelle jedes der n Symbole mit der gleichen Wahrscheinlichkeit $p = \frac{1}{n}$ generiert. Dann nimmt die Entropie ihren maximalen Wert $\log_2(n)$ an:

$$H(X) = -\sum_{i=1}^{n} \frac{1}{n} \log_2(\frac{1}{n}) = -\log_2(\frac{1}{n}) = \log_2 n \tag{3.3}$$

Wird dagegen ein Symbol von der Quelle mit einer Wahrscheinlichkeit $p = 1$ generiert und alle anderen Symbole mit der Wahrscheinlichkeit $p = 0$, dann kann das als nächstes generierte Symbol sicher vorhergesagt werden und es gilt: $H(X) = 0$.
Die gemeinsame Entropie zweier Quellen entspricht dem durchschnittlichen Informationsgewinn, den man erhält, wenn man beide Quellen gleichzeitig beobachtet. Übertragen auf die von ihnen realisierten Zufallsvariablen X und Y ergibt sich für die gemeinsame Entropie:

Definition 3.9 (Gemeinsame Entropie zweier Zufallsvariablen)

$$H(X,Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} \log_2 p_{ij}$$

p_{ij} gibt dabei die Wahrscheinlichkeit für das Ereignis $x_i \vee y_j$ an. Zwischen den beiden Entropien $H(X)$ und $H(Y)$ zweier Zufallsvariablen X und Y sowie ihrer gemeinsamen Entropie $H(X,Y)$ besteht folgender Zusammenhang:

$$H(X,Y) \leq H(X) + H(Y) \tag{3.4}$$

Die Gleichheit gilt genau dann, wenn beide Zufallsvariablen X und Y unabhängig voneinander sind. Damit soll an dieser Stelle mit der wechselseitigen Information (*engl.: mutual information - MI*) ein weiteres zentrales Maß der Informationstheorie eingeführt werden. Die wechselseitige Information beschreibt die gegenseitige Unabhängigkeit zweier Zufallsvariablen X und Y und ist als Differenz zwischen der Summe ihrer beiden Entropien $H(X)$ und $H(Y)$ und der gemeinsamen Entropie $H(X,Y)$ definiert:

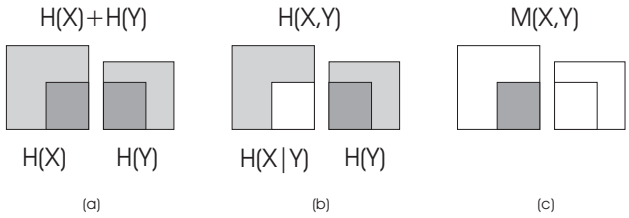


Abbildung 3.4: **Maße der Informationstheorie [36].** Die dunkelgraue Fläche beschreibt die gemeinsame Information von X und Y. Die den Quadraten zugeordneten Maße charakterisieren die Information der jeweils grauen (hell- und dunkelgrau) Flächen.

Definition 3.10 (Wechselseitige Information)

$$MI(X,Y) = H(X) + H(Y) - H(X,Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \log_2 \frac{p_{ij}}{p_i p_j}$$

Sind die beiden Zufallsvariablen X und Y unabhängig, dann hat ihre wechselseitige Information $MI(X,Y)$ den Wert 0. Die wechselseitige Information wird umso größer, je mehr X und Y kovariieren.
Schließlich soll noch eine letzte, wichtige Eigenschaft der wechselseitigen Information vorgestellt werden. Führt man zusätzlich das Maß der bedingten Entropie ein, die die durchschnittliche Restinformation einer Informationsquelle angibt, wenn eine zweite bereits bekannt ist, kann für die gemeinsame Entropie gezeigt werden, daß:

$$H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X) \tag{3.5}$$

Ist eine Zufallsvariable Y beispielsweise vollständig durch eine Zufallsvariable X bestimmt, dann entsteht aus der zusätzlichen Beobachtung ihrer Realisierung kein Informationsgewinn, wenn die Realisierung von X bereits beobachtet wurde. Die bedingte Entropie $H(Y|X)$ nimmt dann den Wert 0 an und für die gemeinsame Entropie $H(X,Y)$ folgt aus Gleichung 3.5:

$$H(X,Y) = H(X) \tag{3.6}$$

Berücksichtigt man diese Gleichheit bei der Definition 3.10 der wechselseitigen Information, ergibt sich:

$$MI(X,Y) = H(Y) \tag{3.7}$$

Um also nachzuweisen, daß ein Element Y vollständig durch ein Element X festgelegt wird, genügt es, eine der Eigenschaften 3.6 bzw. 3.7 zu überprüfen.

Neben der Mathematik, Informatik und Nachrichtenübertragung wird die theoretische Betrachtung der Kommunikation auch zur Beschreibung von Kommunikations-

systemen anderer Bereiche verwendet. Übertragen auf die Domäne der Genregulationsnetzwerke fungiert ein Gen g_i als Sender und Empfänger. Seine Expressionsrate entspricht dann einem vom Sender erzeugten Symbol und damit der Nachricht, die übertragen werden muß. Unterschiedliche Symbole stehen für verschiedene Werte der Expressionsrate. Ein Gen g_j, welches von Gen g_i reguliert wird, hat in seiner Funktion als Empfänger die Aufgabe, die Nachricht von Gen g_i zu analysieren. Als Sender erzeugt g_j anschließend unter Berücksichtigung der empfangenen Nachricht ein neues Symbol, das seiner aktualisierten Expressionsrate entspricht.

Der Algorithmus

In einem Booleschen Netzwerk aus N Booleschen Variablen gibt es für jede Boolesche Variable $\sum_{k=0}^{N} \binom{N}{k} = 2^N$ Mengen potentieller Eltern. Um die Elternmenge $Pa(x_i)$ einer Booleschen Variablen x_i zu bestimmen, wird dieser Suchraum von *Reveal* schrittweise durchlaufen. Dabei betrachtet der Algorithmus zuerst für $k = 1$ alle Mengen, die nur aus einem Inputelement bestehen, danach für $k = 2$ alle zweielementigen Mengen usw., bis er eine Menge X aus k Inputelementen als Elternmenge $Pa(x_i) = \{x_{i1}, x_{i2}, ..., x_{ik}\}$ identifizieren kann. Die Suche wird dann abgebrochen. Ist der Algorithmus bei $k = N$ angelangt und konnte die entsprechende Elternmenge noch nicht identifizieren, ist die Boolesche Variable x_i von unbekannten Größen abhängig und ihre Elternmenge kann nicht bestimmt werden.
Für jede Menge X aus k Inputelementen benutzt *Reveal* die wechselseitige Information $MI(X, x_i)$, um zu berechnen, wieviel Information die Inputelemente auf x_i übertragen. Für diese Berechnung werden die gegebenen Zustandsübergangsdaten bzw. die gegebene Zeitreihe herangezogen – um die Entropien $H(X)$, $H(x_i)$ und $H(X, x_i)$ zu bestimmen, muß man die Wahrscheinlichkeiten p_i für die einzelnen Outputzustände von x_i, die Wahrscheinlichkeiten p_j für die einzelnen Inputzustandsvektoren von X und die Wahrscheinlichkeiten p_{ij} für jegliche Kombinationen der Outputzustände von x_i mit den Inputzustandsvektoren von X aus diesen Trainingsdaten schätzen. Mit der Begründung, daß die Information über die Inputzustände der k Inputelemente von X vollständig auf den Outputzustand der Boolesche Variable x_i übertragen wird, falls es sich bei der Menge X um die Elternmenge von x_i handelt, und x_i somit vollständig von X abhängt, kann man eine der Eigenschaften 3.6 bzw. 3.7 verwenden. Gilt also die Gleichheit

$$MI(X, x_i) = H(x_i) \qquad bzw. \qquad H(X, x_i) = H(X), \tag{3.8}$$

ist mit X die Elternmenge $Pa(x_i)$ der Booleschen Variable x_i eindeutig identifiziert. Anschließend werden alle möglichen Kombinationen der Outputzustände von x_i mit den Inputzustandsvektoren von X in die entsprechende Regeltabelle eingetragen und so die Boolesche Funktion f_i spezifiziert.

Die Gleichheit in 3.8 ist allerdings nur dann nachweisbar, wenn x_i nicht von unbekannten Größen abhängt und die Zustände all seiner Elternelemente in den Trainingsdaten beobachtet wurden. Auch müssen ungestörte Trainingsdaten vorliegen. Da biologische Experimente in der Regel immer gewissen Fehlern unterliegen, und so verrauschte Trainingsdaten entstehen, kann der *Reveal* Algorithmus in dieser ursprünglichen Form nicht angewandt werden, um ein genetisches Netzwerk aus Genexpressionsdaten zu rekonstruieren. In [42, 46] wurde deshalb der *Reveal* Algorithmus entsprechend angepaßt:
Hier wird vorgeschlagen, anstelle der Gleichheitsbeziehung 3.8 mit der Ungleichung

$$MI(X, x_i) > 0 \tag{3.9}$$

zu arbeiten. Ist die wechselseitige Information $M(X, x_i)$ größer null, sind X und x_i voneinander abhängig und die Menge X erklärt die Outputzustände der Booleschen Variablen x_i zwar nicht unbedingt vollständig, aber zumindest teilweise.
Da die benötigten Wahrscheinlichkeiten zur Berechnung der wechselseitigen Information aus den Trainingsdaten geschätzt werden, ist auch die errechnete wechselseitige Information $\hat{MI}(X, x_i)$ nur ein Schätzer der wahren wechselseitigen Information $MI(X, x_i)$. Es reicht deshalb nicht aus zu überprüfen, ob gilt: $\hat{MI}(X, x_i) > 0$, sondern ein statistischer Test muß hinzugezogen werden. Er entscheidet, ob $MI(X, x_i)$ signifikant größer als der Wert 0 ist, oder ob sich die beobachtete Ungleichheit auf zufällige Schwankungen zurückführen läßt. Nach [41] kann die wechselseitige Information zweier Zufallsvariablen X und Y mit der χ^2-Verteilung approximiert werden:

$$\hat{MI}(X, Y) \cdot M \cdot \ln 4 \sim \chi^2(X, Y) \tag{3.10}$$

Die Konstante M beschreibt dabei den Umfang der gegebenen Daten.
Mit Hilfe des χ^2-Unabhängigkeitstests lehnt man die Nullhypothese

$$H_0 : \qquad MI(X, x_i) = 0 \tag{3.11}$$

folglich ab, falls:

$$\hat{MI}(X, x_i) \cdot M \cdot \ln 4 > \chi^2_{f;1-\alpha_1} \tag{3.12}$$

Die Freiheitsgrade f ergeben sich dabei aus $f = (n_{x_i} - 1)(n_X - 1)$ mit n_{x_i} als Anzahl der verschiedenen Outputzustände von x_i und n_X als Anzahl der verschiedenen Inputzustandsvektoren von X.
Ist mit der Ablehnung der Nullhypothese nachgewiesen, daß X und x_i voneinander abhängig sind und die Menge X somit Information auf die Booleschen Variablen x_i überträgt, kann X in die Elternmenge $Pa(x_i)$ von x_i aufgenommen werden, falls X nur aus einem Inputelement besteht. Enthält X allerdings mehr als ein Inputelement, muß der Algorithmus für jedes dieser Inputelemente z zusätzlich kontrollieren, ob es tatsächlich benötigt wird, um die Information auf x_i zu übertragen. Auch hier zieht

er dazu einen statistischen Test heran, der überprüft, ob die Information, welche die Menge X auf x_i überträgt, äquivalent zu der Information ist, die auch die Teilmenge $X\backslash z$ übertragen kann. Um die Nullhypothese

$$H_0 : \qquad MI(X, x_i) = MI(X\backslash z, x_i) \tag{3.13}$$

abzulehnen, wird die in [10] beschriebene Eigenschaft

$$(MI(X, x_i) - MI(X\backslash z, x_i)) \cdot M \cdot \ln 4 \sim \chi^2_{df;1-\alpha_2} \tag{3.14}$$

ausgenutzt und getestet, ob:

$$(\hat{MI}(X, x_i) - \hat{MI}(X\backslash z, x_i)) \cdot M \cdot \ln 4 > \chi^2_{df;1-\alpha_2} \tag{3.15}$$

Die Freiheitsgrade *df* ergeben sich dabei aus $df = f(X, x_i) - f(X\backslash z, x_i)$. Ein positives Testergebnis bedeutet, daß man die Nullhypothese ablehnen kann und die Teilmenge $X\backslash z$ signifikant weniger Information auf x_i überträgt als die Menge X selbst. Damit darf das Inputelement z dann in die Elternmenge $Pa(x_i)$ der Booleschen Variablen x_i aufgenommen werden.
Wie bereits in Abschnitt 3.1.1 beschrieben, geht man beim Prüfen statistischer Hypothesen notwendigerweise das Risiko ein, Fehlentscheidungen zu treffen. Die Signifikanzniveaus α_1 und α_2 müssen deshalb sorgfältig gewählt werden.
Der Suchraum wird äquivalent zur ursprünglichen Form von *Reveal* schrittweise durchsucht. Da der Test 3.9 allerdings nicht nachweist, daß eine Boolesche Variable x_i vollständig von einer Menge X abhängt, kann der Algorithmus nach einem positiven Testergebnis nicht beendet werden. Er muß die Suche nach weiteren Inputelementen der Elternmenge $Pa(x_i)$ fortsetzen. Prinzipiell ist so der gesamte Suchraum zu durchlaufen. Es empfiehlt sich allerdings eine Beschränkung der maximalen Anzahl von Inputelementen in einer Menge X auf einen Wert k_{max} [46]. Zum einen ist eine hohe Konnektivität eines Genregulationsnetzwerks bei großem N biologisch unrealistisch – man geht davon aus, daß ein Gen in der Regel von nicht mehr als acht bis zehn anderen Genen reguliert wird [61]. Zum anderen ist es aufgrund eines meist eher kleinen Umfangs der verfügbaren Daten auch nicht möglich, einen Einfluß einer großen Menge X von Inputelementen auf eine Boolesche Variable x_i mit Hilfe der wechselseitigen Information nachzuweisen, selbst wenn dieser Einfluß tatsächlich existiert.

Implementierung

Für die in [42] angepaßte Variante des *Reveal* Algorithmus läßt sich der folgende Pseudocode angeben:

```
1    FOR i:=1 TO N
2        Pa(x_i) = ∅
```

```
3     FOR i:=1 TO N
4         FOR k:=1 TO k_max
5             FOR EACH possible set X of k inputelements
6                 compute MÎ(X, x_i)
7                 IF MÎ(X, x_i) · M · ln 4 > χ²_{n_{x_i}−1)(n_X−1),1−α_1
8                     IF k=1
9                         add X to Pa(x_i)
10                    ELSE
11                        FOR EACH inputelement z of X
12                            compute MÎ(X\z, x_i)
13                            IF (MÎ(X, x_i) − MÎ(X\z, x_i)) · M · ln 4 > χ²_{df;1−α_2}
14                                IF z ∉ Pa(x_i)
15                                    add z to Pa(x_i)

16    FOR i:=1 TO N
17        create a rule table to specify the Boolean function f_i
```

Im ersten Teil (Zeilen 1-15) wird der Suchraum vom Algorithmus also schrittweise durchlaufen (Zeilen 3-4). Um die Inputelemente einer Menge X als Elternelemente einer Booleschen Variable x_i zu identifizieren, werden die eben beschriebenen Tests durchgeführt und die Elternmenge $Pa(x_i)$ von x_i gegebenenfalls entsprechend erweitert (Zeilen 5-15).
Um die Booleschen Funktionen zu spezifizieren, erzeugt der zweite Teil des Algorithmus (Zeilen 16-17) die jeweiligen Regeltabellen.

Limitationen

Auch für diesen Algorithmus soll abschließend analysiert werden, wie verschiedene Faktoren die Güte der Ergebnisse beeinflussen.
Im Modellierungsprozeß wird ein Genregulationsnetzwerk zu einem diskreten Zeitsystem abstrahiert, in dem alle Gene ihre Expressionsraten synchron aktualisieren. Diese vereinfachende Annahme ist biologisch nicht realistisch. Wie beschrieben resultieren daraus Probleme, wenn es bei der Durchführung der biologischen Experimente darum geht, den Zeitschritt Δt für einen Zustandsübergang festzulegen. Ist dieser zu groß, werden viele Gene des Netzwerks in diesem Zeitraum ihre Expressionsrate mehrfach aktualisieren. Der gemessene Outputzustand des Systems entspricht dann nicht dem direkten Folgezustand des Inputzustands, und somit werden auch indirekte Einflüsse gemessen. Ist der Zeitschritt dagegen zu klein, wird man aufgrund der Tatsache, daß viele Gene für die Aktualisierung ihrer Expressionsraten mehr Zeit als Δt benötigen, viele direkte Einflüsse nicht messen. Dem Algorithmus werden so falsche Informationen zur Verfügung gestellt, denn er erwartet, daß der Outputzustand eines gegebenen Zustandsübergangs direkt aus dem Inputzustand folgt und sich die Expressionsraten aller Gene genau einmal aktualisieren konnten.

Zusätzlich nimmt man außerdem vereinfachend an, daß die Expressionsraten der Gene durch Boolesche Variablen modelliert werden können. Die Trainingsdaten müssen deshalb entsprechend diskretisiert werden, wodurch wichtige Informationen verloren gehen.
Für den zur Identifizierung von Elternmengen herangezogenen χ^2-Unabhängigkeitstest müssen die Signifikanzniveaus α_1 und α_2 sorgfältig ausgewählt werden, denn sie legen in Abhängigkeit vom Stichprobenumfang M auch die Irrtumswahrscheinlichkeiten β_1 und β_2 fest. In diesem Zusammenhang stellt sich außerdem die Frage, inwiefern das Problem des multiplen Testens hier die Güte der Ergebnisse beeinflußt. Es werden mehrere Hypothesen an den gleichen Trainingsdaten getestet; jeweils mit einem bestimmten Signifikanzniveau α_1 bzw. α_2. Die entsprechende gesamte Irrtumswahrscheinlichkeit $\alpha_{1,gesamt}$ bzw. $\alpha_{2,gesamt}$, beim Testen von m Hypothesen mindestens eine wahre Nullhypothese fälschlicherweise abzulehnen, vergrößert sich mit steigender Anzahl m. Es existieren Abschätzungen für diese multiple Irrtumswahrscheinlichkeit (zum Beispiel Bonferroni-Abschätzung [52]: $\alpha_{gesamt} = m \cdot \alpha$), die helfen können, die einzelnen Signifikanzniveaus geeignet festzulegen, wenn die Anzahl der durchzuführenden Tests im voraus bekannt ist. Für den ersten Test, der überprüft, ob eine bestimmte Menge von Inputelementen Information auf ein Outputelement überträgt, steht fest, wie oft er vom Algorithmus ausgeführt wird. Diese Anzahl ist von der Anzahl der Netzwerkkomponenten und von dem Parameter k_{max} abhängig. Das Signifikanzniveau α_1 kann somit unter Berücksichtigung dieser Anzahl geeignet festgelegt werden. Im Gegensatz dazu ist völlig unklar, wie oft der Algorithmus den zweiten Test heranzieht. Diese Anzahl ist davon abhängig, wie oft der erste Test positiv ausfällt. Da man vor der Ausführung des Algorithmus nicht weiß, wie oft dies geschieht, fällt es sehr schwer, die gesamte Irrtumswahrscheinlichkeit $\alpha_{2,gesamt}$ für diesen Test abzuschätzen und das Signifikanzniveau α_2 entsprechend festzulegen. An dieser Stelle ist auch zu berücksichtigen, daß der verwendete χ^2-Unabhängigkeitstest eigentlich einen minimalen Datenumfang von 40 Zustandsübergangspaaren benötigt – diese Anzahl an Zustandsübergängen ist in der Praxis oft nicht verfügbar, was sich zusätzlich negativ auf die Irrtumswahrscheinlichkeiten auswirkt.

3.3 Reverse Engineering in diskreten Dynamischen Bayesschen Netzwerken

Da ein diskretes DBN die kontinuierlichen Expressionsraten der Gene durch diskrete Zufallsvariablen modelliert, muß man die gegebenen Expressionsdaten auch hier entsprechend quantifizieren, bevor sie von einem Reverse Engineering Algorithmus verarbeitet werden können. Im Gegensatz zu den Booleschen Netzwerken können die diskreten Zufallsvariablen in einem Bayesschen Netzwerk auch mehr als zwei Zustände annehmen. Prinzipiell ist die Anzahl der Zustände frei wählbar ; es ist al-

lerdings zu berücksichtigen, daß das Modell umso komplexer wird, je mehr Zustände man verwendet.
Ein diskretes DBN $B = \langle G, \Theta \rangle$ als Modell für ein Genregulationsnetzwerk besteht aus $N(T+1)$ diskreten Zufallsvariablen, die die Expressionsraten der N Gene zu $T+1$ aufeinanderfolgenden Zeitpunkten $0, 1, ..., T$ beschreiben. Die benötigten Trainingsdaten $D = \{d_1, d_2, ..., d_M\}$ bestehen dabei aus M Trainingsvektoren. Jeder Trainingsvektor d_m entspricht einer Trajektorie der Länge $T+1$ des Genregulationsnetzwerks – also einer zeitliche Folge von Systemzuständen des Genregulationsnetzwerks – und repräsentiert damit genau einen Zustand des diskreten DBN:

$$d_m = \begin{pmatrix} x_{1,m}[0] & \cdots & x_{1,m}[T] \\ \vdots & \ddots & \vdots \\ x_{N,m}[0] & \cdots & x_{N,m}[T] \end{pmatrix} \tag{3.16}$$

Aus ihnen muß ein Reverse Engineering Algorithmus die regulatorischen Interaktionen zwischen den Zufallsvariablen, die die Expressionsraten der Gene im Netzwerk modellierten, identifizieren und so die Struktur G des diskreten DBN erlernen. Durch die Schätzung der bedingten Wahrscheinlichkeitsverteilungen werden außerdem die Parameter θ_{i,j_i,k_i} des Systems bestimmt und die Interaktionen genauer spezifiziert. Algorithmen hierfür basieren in der Regel auf einer heuristischen Suche in einem vorher definierten Suchraum möglicher Strukturen und arbeiten mit einer Scoring-Funktion, um ein Bayessches Netzwerk zu lernen, das bezüglich der gegebenen Daten sehr wahrscheinlich ist.
Oftmals sind die Trainingsdaten für einige Variablen nicht vollständig gegeben, weil beispielsweise die entsprechenden Expressionsraten in manchen Expressionsexperimenten nicht gemessen werden konnten. Im Gegensatz zu allen anderen Netzwerkmodellen gibt es für (Dynamische) Bayessche Netzwerke auch Ansätze zur Arbeit mit unvollständigen Trainingsdaten [11, 15, 17]. Darauf soll hier aber nicht weiter eingegangen werden.
Bei der Entwicklung des in dieser Arbeit verwendeten Algorithmus wurden die in [16] und [34] vorgestellten Techniken genutzt:

3.3.1 Lernalgorithmus zur Identifizierung der Struktur eines diskreten DBN

Dieser Algorithmus arbeitet mit der Annahme, daß die Prozesse in dem Genregulationsnetzwerk durch stationäre Markov-Prozesse modelliert werden können. Deshalb müssen, wie in Abschnitt 2.2.3 beschrieben, nur noch Zustandsübergänge des Genregulationsnetzwerks von einem beliebigen Zeitpunkt t zum Folgezeitpunkt $t+1$ betrachtet werden. Der Parameter T hat demzufolge den Wert 1, und das diskrete DBN als Modell des Genregulationsnetzwerks besteht aus den Zufallsvariablen $\mathbf{X}[t] = \{X_1[t], X_2[t], ...X_N[t]\}$, welche die Expressionsraten der Gene vor einem

Zustandsübergang modellieren und aus den Zufallsvariablen $\mathbf{X}[t+1] = \{X_1[t+1], X_2[t+1], ...X_N[t+1]\}$, die den Expressionsraten der Gene nach dem Zustandsübergang entsprechen (Abbildung 2.4(b)). Die als Eingabe erwarteten Trainingsdaten D entsprechen Zustandsübergangsdaten; jeder Trainingsvektor d_m besteht aus einem Ausgangszustand des Genregulationsnetzwerks zu einem Zeitpunkt t und dem Folgezustand zum Zeitpunkt $t+1$:

$$d_m = \begin{pmatrix} x_{i,m}[t] & x_{i,m}[t+1] \\ \vdots & \vdots \\ x_{i,m}[t] & x_{i,m}[t+1] \end{pmatrix} \tag{3.17}$$

Prinzipiell ist es auch möglich, mit einer Zeitreihe zu arbeiten. Es gilt dann: $\forall i : x_{i,m}[t+1] = x_{i,m+1}[t]$.
Das Ziel des Algorithmus ist es nun, mit Hilfe einer Scoring-Funktion einen möglichst guten Schätzer $\hat{G}$ für die wahre Struktur G zu erlernen und so zu identifizieren, wie die Zufallsvariablen in $\mathbf{X}[t+1]$ von den Zufallsvariablen in $\mathbf{X}[t]$ abhängen.

Scoring-Funktionen

Eine Scoring-Funktion $Score(G, D)$ dient zur Berechnung der Güte einer Struktur G bezüglich der gegebenen Trainingsdaten D. Sie stellt somit ein wichtiges Hilfsmittel bei der Suche nach einem guten Schätzer $\hat{G}$ der wahren Struktur G dar.
Ein guter Schätzer zeichnet sich dadurch aus, daß er unter den gegeben Trainingsdaten D sehr wahrscheinlich ist. Deshalb bietet die Posterior-Wahrscheinlichkeit $P(G|D)$ einer Struktur G einen möglichen Ansatz zur Definition einer Scoring-Funktion. Eine Anwendung des Bayesschen Theorems liefert:

$$Score(G|D) = P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{3.18}$$

Die Wahrscheinlichkeit $P(D)$ ist dabei eine strukturunabhängige Konstante und muß nicht weiter betrachtet werden. Mit Hilfe der Prior-Wahrscheinlichkeit $P(G)$ der Struktur ist es möglich, eventuell vorhandenes Vorwissen über die Art der Struktur zu integrieren. Ist kein Vorwissen vorhanden, arbeitet man mit einem nichtinformativen Prior, wodurch sich die Wahrscheinlichkeit $P(G)$ ebenfalls zu einer strukturunabhängigen Konstante reduziert und ignoriert werden kann. Die Wahrscheinlichkeit $P(D|G)$ wird auch marginale Likelihood Funktion genannt. Sie mißt die Wahrscheinlichkeit der gegebenen Daten relativ zu der Struktur G:

$$P(D|G) = \int_0^1 P(D|\Theta_G)P(\Theta_G)d\Theta_G \tag{3.19}$$

Offensichtlich ist es sehr schwierig, die Prior-Wahrscheinlichkeiten $P(\Theta_G)$ der Parametermengen zu einem gegebenen Graphen G zu spezifizieren und das Integral

zu berechnen. Diese aufwendige Berechnung kann umgangen werden, wenn man das asymptotische Verhalten des Terms 3.19 betrachtet. Ein bekannter asymptotischer Schätzer ist das Bayessche Informationskriterium (*engl.: Bayesian information criteria - BIC*) von Schwarz [54]:

$$\log P(D|G) \approx \log P(D|G, \hat{\Theta}_G) - \frac{\log M}{2} dimG \tag{3.20}$$

Die entsprechende Scoring-Funktion, welche genau mit diesem Schätzer arbeitet, wird analog BIC-Score genannt:

Definition 3.11 (BIC-Score [34])

$$Score_{BIC}(G, D) = \log P(D|G, \hat{\Theta}_G) - \tfrac{\log M}{2} dimG$$

Der erste Term entspricht dabei dem Logarithmus der einfachen Likelihood Funktion. Im Gegensatz zur marginalen Likelihood Funktion, die die Wahrscheinlichkeit der gegebenen Daten nur relativ zu der Struktur G bestimmt und so auch die Unsicherheit über die Parameter berücksichtigt, mißt die einfache Likelihood Funktion die Wahrscheinlichkeit der gegebenen Daten relativ zu einer Struktur G und ihrem zugehörigen Maximum Likelihood Schätzer

$$\hat{\Theta}_G = \max_{\Theta_G} L(\Theta_G : D|G) = \max_{\Theta_G} P(D|G, \Theta_G). \tag{3.21}$$

Dieser ist die wahrscheinlichste Parameterinstanz Θ_G von G bezüglich der Trainingsdaten und setzt sich aus den Maximum Likelihood Schätzern $\hat{\theta}_{i,x_i,pa_i}$ der einzelnen Parameter θ_{i,x_i,pa_i} zusammen.
Der Faktor $dimG$ im zweiten Term gibt die Anzahl der Parameter im Modell $\langle G, \hat{\Theta}_G \rangle$ an. Damit kann durch den zweiten Term die Komplexität eines Modells in Abhängigkeit von der Anzahl M der gegebenen Trainingsvektoren bestraft werden. Das Hinzufügen einer Kante wird nur dann mit einem höheren Score belohnt, wenn die resultierende Erhöhung der Likelihood die steigende Komplexität rechtfertigt. Dieser zweite Term wird deshalb auch Strafterm (*engl.: penalty term*) genannt.
Weitere bekannte Scoring-Funktionen sind der Likelihood-Score, der Mutual Information-Score, der Bayessche Score und der Minimum Description Length-Score. Die entsprechenden Definitionen können in [25, 34, 46] nachgelesen werden.

Der Algorithmus

Der Algorithmus durchsucht den Raum S aller relevanten Strukturen nach einer Struktur G, die bezüglich der gegebenen Trainingsdaten D sehr wahrscheinlich ist und so einen guten Schätzer für die wahre Struktur darstellt. Die Güte einer Struktur wird dabei mit Hilfe der eben vorgestellten BIC-Scoring-Funktion bestimmt. Der

Suchraum S umfaßt nur die für das gegebene Problem relevanten Strukturen. Da die Prozesse der Genregulation bei der Modellierung zu stationären Markov-Prozessen vereinfacht werden, soll das Modell $B = \langle G, \Theta \rangle$ des Genregulationsnetzwerks beschreiben, wie die Expressionsraten der Gene zu einem Zeitpunkt $t + 1$ durch die Expressionsraten zum vorangegangen Zeitpunkt t festgelegt werden. Deshalb dürfen die betrachteten Strukturen nur Kanten von den Zufallsvariablen $\mathbf{X}[t]$ zu den Zufallsvariablen $\mathbf{X}[t+1]$ enthalten. Insbesondere sind die Elternmengen der Zufallsvariablen $\mathbf{X}[t]$ leer.
Die Suche nach dem optimalen, bei gegeben Trainingsdaten D am wahrscheinlichsten Schätzer $\hat{G}$ ist NP-hart, da die Anzahl möglicher Strukturen super-exponentiell mit der Anzahl der Variablen wächst [9]. Deshalb muß auf heuristische Suchalgorithmen, zum Beispiel Greedy-Hill-Climbing oder Simulated-Annealing, zurückgegriffen werden.
Der hier vorgestellte Lernalgorithmus basiert auf der Greedy-Hill-Climbing Strategie. Diese beginnt die Suche mit einer bestimmten Startstruktur, bei der es sich um die leere Struktur, aber auch um eine zufällig gewählte Struktur handeln kann. In jedem Lernschritt versucht der Algorithmus eine neue Struktur auszuwählen, die einen höheren Score liefert als die aktuelle Struktur G_{akt}. Dazu werden alle Nachbarstrukturen G_{nb} von G_{akt} evaluiert. Die Menge der Nachbarstrukturen $Nb(G_{akt})$ von G_{akt} ist eine Teilmenge des Suchraums S und besteht aus all den Strukturen, die sich von G_{akt} in nur einer Kante unterscheiden. Sie enthält also jede Struktur von S, die aus G_{akt} durch eine der drei Operationen

1. Einfügen einer Kante
2. Löschen einer Kante
3. Drehen einer Kante

gebildet werden kann. Wegen der Einschränkung des Suchraums S auf die oben beschriebenen relevanten Strukturen entsteht durch das Drehen einer Kante niemals eine bezüglich S gültige Struktur; die dritte Operation entfällt deshalb bei dieser konkreten Anwendung des Algorithmus. Aus dem gleichen Grund ist zu beachten, daß bei der ersten Operation nur durch das Einfügen einer Kante von einer Zufallsvariablen in $\mathbf{X}[t]$ zu einer Zufallsvariablen in $\mathbf{X}[t+1]$ eine gültige Struktur entstehen kann. Für jede Struktur in $Nb(G_{akt})$ berechnet der Algorithmus anschließend den BIC-Score, um die Nachbarstruktur $G_{nb,max}$ mit dem höchsten Score zu bestimmen. Gibt es mehrere Nachbarstrukturen mit dem gleichen maximalen Score, wählt er eine von ihnen zufällig aus. Diese Nachbarstruktur wird dann, falls ihr Score größer oder gleich dem Score der aktuellen Struktur G_{akt} ist, zur aktuellen Struktur des nächsten Lernschrittes ernannt. Im anderen Fall bricht die Suche ab, und die aktuelle Struktur G_{akt} mit dem entsprechenden Maximum Likelihood Schätzer $\hat{\Theta}_{G_{akt}}$ bildet das Ergebnis der Suche.

Ein Problem dieser Greedy-Hill-Climbing Strategie stellen lokale Maxima dar. Ist die aktuelle Struktur G_{akt} lokal optimal, dann liefern alle ihre Nachbarstrukturen einen schlechteren Score, der Algorithmus beendet die Suche und wird so den globalen optimalen Schätzer $\hat{G}$ der wahren Struktur nicht aufspüren. In [34] ist folgender Verbesserungsvorschlag (Random-restart) zu finden: Hat der Algorithmus das Abbruchkriterium erreicht, wird die aktuelle Struktur G_{akt} als Zwischenergebnis gespeichert und anschließend durch das zufällige Einfügen und Löschen einer begrenzten Anzahl von Kanten zur Erzeugung einer neuen Startstruktur verwendet. Die Suche beginnt dann erneut. Nachdem dieser Prozeß n mal wiederholt wurde, wird als endgültiges Ergebnis das Zwischenergebnis mit dem höchsten Score ausgewählt.
Ein weiteres Problem ergibt sich, wenn der Algorithmus auf ein Plateau der Scoring-Funktion – also auf eine Menge benachbarter Strukturen, die alle den gleichen Score besitzen – stößt, denn die Suche erfolgt dann nur noch zufällig. Abhilfe kann hier dadurch erfolgen, daß die aktuellen Strukturen der l vorangegangenen Lernschritte in einer Liste gespeichert und im augenblicklichen Lernschritt nicht ausgewählt werden dürfen (TABU-search [34]).
Schließlich soll noch kurz auf die Berechnung des BIC-Scores einer Struktur G eingegangen werden. Für diese Berechnung ist es im wesentlichen notwendig, die Likelihood der Struktur G und ihrer wahrscheinlichsten Parameterinstanz $\hat{\Theta}_G$ zu bestimmen:

$$L(\langle G, \hat{\Theta}_G \rangle : D) = P(D|G, \hat{\Theta}_G) = \prod_{m=1}^{M} P(d_m|G, \hat{\Theta}_G) \tag{3.22}$$

Jeder gegebene Trainingsvektor d_m repräsentiert genau einen Zustand des Bayesschen Netzwerks. Die Verbundwahrscheinlichkeitsverteilung (2.4) kann benutzt werden, um die Wahrscheinlichkeit des Auftretens von d_m in dem Bayesschen Netzwerk $B = \langle G, \hat{\Theta}_G \rangle$ zu errechnen :

$$P(d_m|G, \hat{\Theta}_G) = \prod_{m=1}^{M} \prod_{j=1}^{N} P(x_{j,m}(t)|pa_{j,m}) \cdot \prod_{i=1}^{N} P(x_{i,m}(t+1)|pa_{i,m}) \tag{3.23}$$

Aufgrund der Einschränkung des Suchraums S auf die relevanten Strukturen sind die Elternmengen der Zufallsvariablen in $\mathbf{X}[t]$ für jeder Struktur $G \in S$ leer und es gilt $\prod_{j=1}^{N} P(x_{j,m}(t)|pa_{j,m}) = \prod_{j=1}^{N} P(x_{j,m}(t))$. Dieser Term reduziert sich damit zu einer strukturunabhängigen Konstanten und kann bei der Berechnung des BIC-Scores ignoriert werden, denn er trägt nicht mehr zur Unterscheidung der einzelnen Strukturen in S bei. Für die Likelihood einer Struktur G und der Instanz $\hat{\Theta}_G$ ergibt sich somit:

$$L(\langle G, \hat{\Theta}_G \rangle : D) \propto \prod_{i=1}^{N} \prod_{m=1}^{M} P(x_{i,m}(t+1)|pa_{i,m}) \tag{3.24}$$

Sei N_{i,j_i,k_i} die Anzahl der Trainingsvektoren, bei denen die Zufallsvariable $X_i[t+1]$ den Zustand k_i und ihre Elternmenge $Pa(X_i[t+1])$ den Zustandsvektor j_i aufweisen. Die Wahrscheinlichkeit $P(X_i[t+1] = k_i|Pa(X_i[t+1]) = j_i) = \theta_{i,j_i,k_i}$ kann aus den Trainingsdaten mit Hilfe der Maximum Likelihood Schätzung bestimmt werden. Der Maximum Likelihood Schätzer $\hat{\theta}_{i,j_i,k_i}$ ergibt sich aus der Maximierung der Likelihoodfunktion $L(\Theta_G : D|G)$ bezüglich θ_{i,j_i,k_i} und entspricht der in den Trainingsdaten D beobachteten relativen Häufigkeit $\frac{N_{i,j_i,k_i}}{N_{i,j_i}}$ mit $N_{i,j_i} = \sum_{k_i} N_{i,j_i,k_i}$ [16]:

$$\begin{aligned} L(\langle G, \hat{\Theta}_G \rangle : D) &\propto \prod_{i=1}^{N} \prod_{j_i} \prod_{k_i} P(X_i[t+1] = k_i|Pa(X_i[t+1]) = j_i)^{N_{i,j_i,k_i}} \\ &= \prod_{i=1}^{N} \prod_{j_i} \prod_{k_i} \hat{\theta}_{i,j_i,k_i}^{N_{i,j_i,k_i}} \\ &= \prod_{i=1}^{N} \prod_{j_i} \prod_{k_i} \frac{N_{i,j_i,k_i}}{N_{i,j_i}}^{N_{i,j_i,k_i}} \end{aligned} \tag{3.25}$$

Sei s_i die Anzahl der möglichen Zustände einer Variablen $X_i[t]$ und q_i die Anzahl der möglichen Zustandsvektoren ihrer Elternmenge[4]. Dann benötigt man genau $q_i \cdot s_i$ Parameter, um die bedingte Wahrscheinlichkeitsverteilung von $X_i[t+1]$ zu beschreiben. Von ihnen sind $q_i(s_i - 1)$ frei wählbar. Damit kann die Komplexität der Struktur G durch die Summe $\sum_{i=1}^{N} q_i(s_i - 1)$ ausgedrückt werden [16]. Auch hier müssen die Zufallsvariablen in $\mathbf{X}[t]$ nicht berücksichtigt werden – da ihre Elternmengen in jeder betrachteten Struktur leer sind, ist die Anzahl der Parameter zur Spezifizierung ihrer bedingten Wahrscheinlichkeitsverteilungen strukturunabhängig.
Zusammenfassend ergibt sich für den BIC-Score einer Struktur G die folgende Formel:

$$\begin{aligned} Score_{BIC}(G, D) &= \log P(D|G, \hat{\Theta}_G) - \frac{\log M}{2} dim G \\ &\propto \log \prod_{i=1}^{N} \prod_{j_i} \prod_{k_i} \frac{N_{i,j_i,k_i}}{N_{i,j_i}}^{N_{i,j_i,k_i}} - \frac{\log M}{2} \cdot \sum_{i=1}^{N} q_i(s_i - 1) \end{aligned} \tag{3.26}$$

$$= \sum_{i=1}^{N} \sum_{j_i} \sum_{k_i} N_{i,j_i,k_i} \log \frac{N_{i,j_i,k_i}}{N_{i,j_i}} - \frac{\log M}{2} \cdot \sum_{i=1}^{N} q_i(s_i - 1) \tag{3.27}$$

[4]Kann jede Zufallsvariablen genau z verschiedene Zustände annehmen und besteht die Elternmenge $Pa(X_i[t+1])$ aus k Variablen, dann gilt: $s_i = z$ und $q_i = z^k$.

Der größte Rechenaufwand entsteht bei der Evaluation der Nachbarstrukturen in jedem Lernschritt, und somit beeinflußt dieser Schritt wesentlich die Laufzeit des Algorithmus. Um die Berechnung des Scores einer Nachbarstruktur möglichst effizient zu gestalten, nutzt der Algorithmus hierbei aus, daß sich der BIC-Score als ein Produkt (Summe) von einzelnen Wahrscheinlichkeiten ausdrücken läßt, die jeweils nur von einer Zufallsvariablen und deren Eltern abhängt (Gleichungen 3.26, 3.27). Diese wichtige Eigenschaft einer Scoring-Funktion heißt Zerlegbarkeit [25]:

Definition 3.12 (Zerlegbarkeit) *Kann ein Score als ein Produkt einzelner Terme ausgedrückt werden, die jeweils nur von einer Zufallsvariablen und deren Eltern abhängen, dann ist dieser Score zerlegbar.*

Bei der Berechnung des Scores einer Nachbarstruktur G_{nb} kann deshalb der Score der aktuellen Struktur G_{akt} herangezogen werden. Der Algorithmus muß jeweils nur den Term der Zufallsvariablen neu berechnen, deren Elternmenge sich bei der Konstruktion der Nachbarstruktur verändert hat.

Implementierung

Der Pseudocode für den Lernalgorithmus lautet wie folgt:

```
1   Choose start structure G_start
2   FOR i:=1 TO n
3       G_max := G_start
4       score_{G_max} :=COMPUTE_SCORE(G_max)
5       DO
6           G_akt := G_max
7           score_{G_akt} := score_{G_max}
8           score_{G_max} := -∞
9           FOR EACH node X_j[t] in G_akt
10              FOR EACH node X_i[t+1] in G_akt
11                  G_nb := G_akt
12                  IF edge e = (X_j[t], X_i[t+1]) ∈ G_nb
13                      delete e in G_nb
14                  ELSE
15                      add e to G_nb
16                  IF G_nb has not been chosen during the last l learning steps
17                      score_{G_nb} :=UPDATE_SCORE(G_nb, G_akt, score_akt, i)
18                      IF score_{G_nb} > score_{G_max}
19                          G_max := G_nb
20                          score_{G_max} := score_{G_nb}
21      WHILE score_{G_max} ≥ score_{G_akt}
22      IF i==1
23          G_save := G_akt
24          score_{G_save} := score_{G_akt}
25      ELSE
```

26 **IF** $score(G_{save}) < score(G_{akt})$
27 $G_{save} := G_{akt}$
28 $score_{G_{save}} := score_{G_{akt}}$
29 $G_{start} := change\_randomly(G_{akt})$
30 **RETURN** $\langle G_{save}, \hat{\Theta}_{G_{save}} \rangle$

31 **PROCEDURE** COMPUTE_SCORE(G)
32 $score := 0$
33 **FOR EACH** *node* $X_i[t+1]$ *in* G
34 $score+ = \prod_{j_i} \prod_{k_i} \frac{N_{i,j_i,k_i}}{N_{i,j_i}}^{N_{i,j_i,k_i}}$
35 $score := \log(score) - \frac{\log M}{2} \cdot dimG$
36 **RETURN** *score*

37 **PROCEDURE** UPDATE_SCORE($G_{nb}, G_{akt}, score_{akt}, i$)
38 $score_{G_{nb}} = score_{G_{akt}}$
39 $score_{G_{nb}}+ = \frac{\log M}{2} dimG_{akt}$
40 $score_{G_{nb}}- = \log \prod_{j_{i,akt}} \prod_{k_i} \frac{N_{i,j_{i,akt},k_i}}{N_{i,j_{i,akt}}}^{N_{i,j_{i,akt},k_i}}$
41 $score_{G_{nb}}+ = \log \prod_{j_{i,nb}} \prod_{k_i} \frac{N_{i,j_{i,nb},k_i}}{N_{i,j_{i,nb}}}^{N_{i,j_{i,nb},k_i}}$
42 $score_{G_{nb}}- = \frac{\log M}{2} dimG_{nb}$
43 **RETURN** $score_{G_{nb}}$

Ausgehend von einer Startstruktur (Zeilen3-4) werden in jedem Lernschritt alle möglichen Nachbarstrukturen der aktuellen Struktur G_{akt} generiert (Zeilen 9-15). Die Berechnung des Scores einer Nachbarstruktur erfolgt in der Prozedur *UPDATE_SCORE*, welche die Zerlegbarkeit der BIC-Scoring-Funktion ausnutzt und den Score $score_{G_{nb}}$ der jeweiligen Nachbarstruktur G_{nb} durch Addition und Subtraktion der entsprechenden Terme aus dem Score $score_{G_{akt}}$ der aktuellen Struktur G_{akt} errechnet (Zeilen 37-43). Die Nachbarstruktur mit dem höchsten Score wird als aktuelle Struktur des nächsten Lernschrittes ausgewählt (Zeilen 6-7). Um der zufälligen Suche des Algorithmus auf einem Plateau entgegenzuwirken, betrachtet man immer nur die Nachbarstrukturen, die verschieden zu den aktuellen Strukturen der letzten l Lernschritte sind (TABU-search – Zeile 16). Das Greedy-Hill-Climbing bricht ab, wenn mit keiner der Nachbarstrukturen ein Score erzielt werden kann, der mindestens genauso groß ist wie der Score der aktuelle Struktur G_{akt} (Zeile 21). Anschließend wird die aktuelle Struktur G_{akt} gegebenenfalls als Zwischenergebnis gespeichert (Zeilen 22-28) und zur Erzeugung einer neuen Startstruktur G_{start} durch das zufällige Einfügen und Löschen von Kanten (Zeile 29) benutzt. Der Algorithmus startet das Greedy-Hill-Climbing dann erneut (Random-restart). Insgesamt wiederholt er diesen Prozeß n mal (Zeile 2).

Limitationen

Bei diesem Algorithmus sind es im wesentlichen die vereinfachten Modellannahmen – also modellbedingte Limitationen –, die die Güte der Ergebnisse negativ beeinflussen:
Zum einen arbeitet dieser Lernalgorithmus, genau wie der *Reveal* Algorithmus, mit der vereinfachenden, aber biologisch unrealistischen Annahme, daß ein Genregulationsnetzwerk durch ein diskretes, synchrones Zeitsystem modelliert werden kann. Je nach Wahl des Zeitschrittes Δt der zwischen dem Anfangszustand und dem Folgezustand eines Zustandübergangspaares liegt, repräsentiert ein Trainingsvektor d_m nicht nur – wie vom Algorithmus erwartet – direkte, sondern auch indirekte Interaktionen, wenn es Gene gibt, die ihre Expressionsrate mehr als einmal in diesem Zeitraum aktualisieren. Existieren auf der anderen Seite Gene, die für eine Aktualisierung ihrer Expressionsrate länger als Δt benötigen, können einige direkte Interaktionen in d_m nicht beobachtet werden. Der Algorithmus arbeitet dann mit falschen Informationen.
Zum anderen wirkt sich auch hier die Diskretisierung der Expressionsraten negativ auf die Ergebnisse aus, denn dadurch gehen wichtige Informationen verloren und das Modell verliert an Genauigkeit. Wesentlich ist hier vor allem die Wahl der Intervallgröße, die zur Quantifizierung der kontinuierlichen Expressionsraten benutzt wird. Eine Verkleinerung der Intervalle führt zu weniger Informationsverlust, aber auch zu komplexeren Modellen mit vielen Parametern.
Eine algorithmusbedingte Limitation ergibt sich aus der Tatsache, daß der Algorithmus trotz Random-restart-Strategie bei der Suche nach einem guten Schätzer für die Struktur G nicht unbedingt die Struktur mit dem global maximalen Score findet und dann nur ein lokales Optimum liefert.

3.4 Reverse Engineering in Additiven Regulationsmodellen

Zur Modellierung der Expressionsrate eines Gens dient in diesem Modellansatz eine kontinuierliche Variable. Die zeitliche Änderung einer Variablen wird mit Hilfe einer Differentialgleichung beschrieben, die alle regulierenden Einflüsse additiv berücksichtigt. Nach Einführung von diskreten Zeitschritten ist es auch möglich, die zeitliche Entwicklung einer Variablen durch eine Aktualisierungsregel zu beschreiben:

linearer Ansatz:

$$x_i(t+1) = \sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i \tag{3.28}$$

nichtlinearer Ansatz:

$$x_i(t+1) = \frac{max_i}{1 + e^{-(\sum_{j=1}^{N} w_{ij} x_j(t) + \beta_i)}} \tag{3.29}$$

Die genspezifischen Konstanten β_i können zur Vereinfachung der Aktualisierungsregeln in die Gewichtsmatrix integriert werden. Dazu wird die Gewichtsmatrix W um eine zusätzliche Spalte erweitert, deren Einträge $w_{i0} = \beta_i$ die Biasfaktoren der Gene enthalten.
Die Aufgabe eines Reverse Engineering Algorithmus ist es nun, mit Hilfe der gegebenen Trainingsdaten D einen möglichst guten Schätzer $\hat{W}$ für die wahre Gewichtsmatrix

$$W = \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \end{pmatrix} = \begin{pmatrix} w_{10} & w_{11} & \cdots & w_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N0} & w_{N1} & \cdots & w_{NN} \end{pmatrix} \tag{3.30}$$

zu bestimmen.
Die Algorithmen auf diesem Gebiet arbeiten in der Regel mit Zustandsübergangsdaten oder Zeitreihen vom Datenumfang M, die in folgender Form aufbereitet werden können:

Inputmatrix:

$$U = \begin{pmatrix} \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^M \end{pmatrix} = \begin{pmatrix} 1 & u_1^1 & \cdots & u_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & u_1^M & \cdots & u_N^M \end{pmatrix} \tag{3.31}$$

Outputmatrix:

$$Y = \begin{pmatrix} \mathbf{y}^1 \\ \vdots \\ \mathbf{y}^M \end{pmatrix} = \begin{pmatrix} y_1^1 & \cdots & y_N^1 \\ \vdots & \ddots & \vdots \\ y_1^M & \cdots & y_N^M \end{pmatrix} \tag{3.32}$$

Jeder Zeilenvektor $\mathbf{u}^m$ entspricht dabei einem Inputzustand des Netzwerks, der die Expressionsraten der Gene zu einem Zeitpunkt t beschreibt. Der entsprechende Zeilenvektor $\mathbf{y}^m$ gibt dann die aktualisierten Expressionsraten zum Zeitpunkt $t+1$ an und entspricht so dem zum Inputzustand gehörenden Outputzustand des Netzwerks. Die Spaltenvektoren $\mathbf{u}_i$ und $\mathbf{y}_i$ der Matrizen enthalten alle in den Daten beobachteten Expressionsraten eines Gens g_i jeweils vor und nach den Zustandsübergängen. Der zusätzliche Spaltenvektor $\mathbf{u}_0$ der Inputmatrix ist aufgrund der Erweiterung der Gewichtsmatrix notwendig, um den konstanten Input vom Wert 1 für die jeweiligen Biasfaktoren zu modellieren.

Arbeitet man mit Zustandsübergangsdaten, kommt der Zeilenvektor $\mathbf{u}^m$ dem Anfangszustand eines Zustandsübergangspaares gleich und der entsprechende Zeilenvektor $\mathbf{y}^m$ dem zugehörigen Folgezustand. Ist stattdessen eine Zeitreihe gegeben, entspricht der Outputzustand des Systems nach einem Zustandsübergang dem Inputzustand des Systems vor der nächsten Aktualisierung der Expressionsraten, und so gilt: $\forall m : \mathbf{y}^m = \mathbf{u}^{m+1}$.
Im nichtlinearen Ansatz ergibt sich der Folgezustand einer Variablen x_i durch die Anwendung einer sigmoiden Funktion auf den regulatorischen Input r_i. Um analog zum linearen Ansatz einen linearen Zusammenhang zwischen dem Inputzustand und dem zugehörigen Outputzustand herzustellen, muß auf die einzelnen Werte von Y die entsprechende Umkehrfunktion der Sigmoidalfunktion angewandt werden:

$$y_i^m = -\ln(\frac{max_i}{y_i^m} - 1) \tag{3.33}$$

Damit kann die Problemstellung allgemein – also unabhängig vom linearen oder nichtlinearen Ansatz – auch durch folgende Gleichungen beschrieben werden:

$$Y = UW^T \qquad \text{bzw.} \qquad \forall_i \;\; \mathbf{y}_i = U\mathbf{w}_i^T \tag{3.34}$$

Gesucht ist die Gewichtsmatrix W, die diese Gleichungen erfüllt. $\mathbf{w}_i$ entspricht hier einem Zeilenvektor der Gewichtsmatrix W, der die Gewichte aller regulatorischen Einflüsse auf x_i spezifiziert.
Im allgemeinen geht man bei diesen Reverse Engineering Algorithmen von einem vollständig verknüpften Netzwerk aus. Die wahre Struktur des Netzwerks wird dann nur implizit über den Schätzer $\hat{W}$ der Gewichtsmatrix W festgelegt – ergibt sich für den Schätzer w_{ij} ein Wert 0, dann übt die Expression von Gen g_j keinen regulatorischen Einfluß auf die Expression von Gen g_i aus. Typisch für die Struktur von vor allem größeren genetischen Netzwerken ist eine eher kleine Konnektivität in Bezug auf die Anzahl N der im Netzwerk involvierten Gene. Die zugehörige Gewichtsmatrix W ist somit meist dünn besetzt, d.h. viele ihrer Einträge entsprechen dem Wert 0. Die fehlende Beobachtung wichtiger, unbekannter Einflußfaktoren sowie Meßfehler und Inkonsistenzen in den Daten machen es allerdings schwer, den wahren Wert 0 solcher Gewichte zu schätzen. Die entsprechenden Schätzer nehmen zwar meist einen kleinen Wert an, aber selten direkt den Wert 0. Für die Identifizierung der Struktur ist es deshalb notwendig, die berechneten Schätzer $\hat{w}_{ij}$ anschließend aufgrund ihrer Größe durch die Einteilung in die Klassen „null“ (kein Einfluß von x_j auf x_i) und „nicht-null“ (Einfluß von x_j auf x_i) zu klassifizieren.
Auf dem Gebiet der Additiven Regulationsmodelle sind viele, zum Teil auf recht unterschiedlichen Strategien basierende Ansätze für einen Reverse Engineering Algorithmus zu finden. Drei von ihnen – ein analytisches Verfahren (*Reverse Engineering in Matrizen (REM)*), ein stochastisches, an der Natur orientiertes Optimierungsver-

fahren (*evolutionärer Algorithmus*), sowie ein deterministisches, gradientenbasiertes Optimierungsverfahren (*Backpropagation through time (BPTT)*) – werden hier näher vorgestellt:

3.4.1 REM - Reverse Engineering in Matrizen (Weaver et al. [64])

Dieser erste Algorithmus benutzt die lineare Algebra, um bezüglich der gegebenen Trainingsdaten D einen möglichst guten Schätzer $\hat{W}$ zu finden. Mit Hilfe der Methode der kleinsten Quadrate wird der Gewichtsvektor $\mathbf{w}_i$ einer Variablen x_i so geschätzt, daß die Summe der Fehlerquadrate (euklidischer Fehler)

$$e = \|\mathbf{y}_i - U\hat{\mathbf{w}}_i^T\| = \sum_{m=1}^{M}(y_i^m - \sum_{j=0}^{N} u_j^m w_{ij})^2 \tag{3.35}$$

minimiert wird. Prinzipiell impliziert dies nur die Lösung des Gleichungssystems:

$$\hat{\mathbf{w}}_i^T = U^{-1}\mathbf{y}_i \tag{3.36}$$

Da der Gewichtsvektor $\mathbf{w}_i$ die Dimension $N+1$ besitzt, ist das Gleichungssystem aber unterbestimmt, falls der Umfang M der gegebenen Trainingsdaten kleiner als diese Dimension ist. Gilt dagegen : $M > N+1$, ist das Gleichungssystem überbestimmt. Die Methode der kleinsten Quadrate arbeitet deshalb mit der Pseudoinversen der Matrix U:

$$\hat{\mathbf{w}}_i^T = (U^T U)^{-1} U^T \mathbf{y}_i \tag{3.37}$$

Als Eingabe erwartet der Algorithmus einen Trainingsdatensatz und einen Testdatensatz. Da für jede Variable x_i genau $N+1$ Parameter w_{ij} geschätzt werden müssen, enthält der Trainingsdatensatz idealer Weise mindestens $N+1$ Zustandsübergänge. Für den Testdatensatz sind zwei Zustandsübergangspaare ausreichend; er dient dem Algorithmus zur Berechnung der Güte eines geschätzten Gewichtsvektors $\hat{\mathbf{w}}_i$.

Der Algorithmus

Arbeitet der Algorithmus mit dem nichtlinearen Ansatz, so muß er zuerst für jedes Gen g_i die maximale Expressionsrate max_i bestimmen, um durch die Umformung aller Werte y_i^m nach 3.33 den linearen Zusammenhang zwischen Input- und Outputmatrix herzustellen. Dabei wird angenommen, daß die maximalen Expressionsraten empirisch aus den gegebenen Daten geschätzt werden können und so die größte, in den Daten beobachtete Expressionsrate eines Gens g_i zur Bestimmung von max_i herangezogen werden kann. Es ist zu beachten, daß max_i immer größer als die

größte, beobachtete Expressionsrate von g_i sein muß, denn die Expressionsrate eines Gens läuft im nichtlinearen Ansatz nur asymptotisch für einen unendlich großen regulatorischen Input gegen die maximale Expressionsrate max_i. In der für diese Arbeit implementierten Realisierung bekommt max_i den kleinsten ganzzahligen Wert zugewiesen, der größer ist als alle in den Trainings- und Testdaten beobachteten Expressionsraten von Gen g_i.
Um den Gewichtsvektor $\mathbf{w}_i$ der Variablen x_i zu schätzen, werden folgende Schritte iterativ wiederholt, solange die Dimension von $\hat{\mathbf{w}}_i$ größer null ist:

1. Bestimme $\hat{\mathbf{w}}_i^T = (U_{train}^T U_{train})^{-1} U_{train}^T \cdot \mathbf{y}_{i,train}$ aus den Trainingsdaten.
2. Berechne $e = \|\mathbf{y}_{i,test} - U_{test}\hat{\mathbf{w}}_i^T\|$ bezüglich der Testdaten.
3. Speichere $\hat{\mathbf{w}}_i^T$ und e als Zwischenergebnis.
4. Nutze die Annahme, daß viele Gewichte vom Wert 0 sein sollten; bestimme das Gewicht $\hat{w}_{ij}^{min}$ mit dem betragsmäßig kleinsten Wert und lösche den entsprechenden Spaltenvektor $\mathbf{u}_j$ der Inputmatrix.

Als endgültige Lösung wird anschließend das Zwischenergebnis mit dem kleinsten euklidischen Fehler e ausgewählt.

Implementierung

Es folgt der Pseudocode für den beschriebenen Algorithmus:

```
1   FOR i:=1 TO N
2       find max_i in u_{i,train}, u_{i,test}, y_{i,train}, y_{i,test}
3       FOR m:=1 TO M_train
4           y^m_{i,train} = -ln(max_i / y^m_{i,train} - 1)
5       FOR m:=1 TO M_test
6           y^m_{i,test} = -ln(max_i / y^m_{i,test} - 1)

7   Ŵ : Matrix[N][N+1]
8   FOR i:=1 TO N
9       FOR j:=0 TO N
10          weightIdentifier[j] := j

11      e^min := ∞
12      FOR j:=1 TO N+1
13          compute ŵ^T := (U^T_train U_train)^{-1} U^T_train y_{i,train}
14          compute e = ||y_{i,test} - U_test ŵ^T||
15          IF e < e^min
16              e^min := e
17              ŵ^min := ŵ
18              weightIdentifier^min := weihtIdentifier
```

```
19              weight_min := ∞
20              FOR k:=0 TO N-j+1
21                  IF weight_min > |ŵ[k]|
22                      weight_min := |ŵ[k]|
23                      minIndex := k
24              delete column u_minIndex
25              delete weightIdentifier[minIndex]

26          FOR j:=0 TO N
27              Ŵ[i][j] := 0
28          FOR j:=0 TO ŵ^min.length − 1
29                  Ŵ[i][weightIdentifier[j]] := ŵ^min[j]
```

Der erste Teil (Zeilen 1-6) dient zur Identifizierung der maximalen Expressionsraten und der entsprechenden Umformung der Werte der Outputmatrix. Er entfällt, wenn man mit dem linearen Ansatz arbeitet.
Der zweite Teil (Zeilen 7-29) implementiert die iterative Schätzung des entsprechenden Gewichtsvektors $\mathbf{w}_i$ einer jeden Variablen x_i. Beginnend mit der vollständigen Inputmatrix U wird in jeder Iteration der Gewichtsvektor mit der Methode der kleinsten Quadrate neu berechnet (Zeile 13) und als Zwischenergebnis gespeichert, falls er einen kleineren Fehler liefert als alle anderen Gewichtsvektoren vor ihm (Zeilen 15-18). Anschließend bestimmt der Algorithmus das betragsmäßig kleinste Gewicht des eben neu berechneten Gewichtsvektors $\hat{\mathbf{w}}_i$ (Zeilen 19-23). Die Expressionsraten der Variable, deren Einfluß auf x_i durch dieses kleinste Gewicht beschrieben wird, werden aus der Inputmatrix gestrichen (Zeile 24). Ebenfalls kann in dem Vektor *weightIdentifier*, der für jedes verbleibende Gewicht w_{ij} den entsprechenden Index j enthält und so die Zuordnung der Werte des geschätzten Gewichtsvektors zu den Einträgen der Gewichtsmatrix erlaubt, der Index dieses kleinsten Gewichts gelöscht werden (Zeile 25). Anschließend beginnt die Berechnung des Gewichtsvektors erneut. Am Ende trägt der Algorithmus das Zwischenergebnis mit dem kleinsten euklidischen Fehler bezüglich der Testdaten als endgültige Lösung in die Gewichtsmatrix $\hat{W}$ ein (Zeile 26-29).

Limitationen

Die Limitationen dieses Reverse Engineering Algorithmus entstehen im wesentlichen durch vereinfachte, aber biologisch unrealistische Annahmen bei der Modellierung eines Genregulationsnetzwerks mit einem Additiven Regulationsmodell:
So wird beispielsweise auch hier das Genregulationsnetzwerk zu einem synchronen, diskreten Zeitsystem vereinfacht, woraus bereits beschriebene Probleme resultieren[5]. Des weiteren werden die regulatorischen Interaktionen als unabhängige Ereignisse

[5]Siehe Abschnitt *Limitationen* in 3.2.1.

betrachtet, die additiv zusammenwirken. Demgegenüber steht die experimentell bewiesene Tatsache, daß es Gene gibt, deren Expression nur durch eine bestimmte Kombination von Transkriptionsfaktoren reguliert werden kann. Ein einzelner dieser Transkriptionsfaktoren kann ohne die Anwesenheit der anderen keinen Einfluß nehmen.
Das Additive Regulationsmodell arbeitet mit deterministischen Beziehungen und ist damit nicht in der Lage, aufgrund von Fehlern und Inkonsistenzen stochastisch erscheinende Beziehungen zu modellieren. Deshalb hängt die Güte der Ergebnisse auch maßgeblich von der Qualität der gegebenen Daten ab. Je weniger Meßfehler in den Daten enthalten sind, desto genauer kann der Schätzer $\hat{W}$ der Gewichtsmatrix bestimmt werden und desto korrekter lassen sich die Gewichte klassifizieren.
Die vereinfachende Betrachtung der Genregulationsprozesse – also die Einschränkung der Regulationsprozesse auf die Ebene der Transkription, die Annahme einer starken Korrelation zwischen mRNA- und Protein-Konzentrationen eines Gens und die Modellierung der Expressionsrate eines Gens allein durch die Konzentration seines mRNA-Transkripts – kommt bei den Algorithmen für diesen Modellansatz besonders nachteilig zum Tragen: Im Reverse Engineering Prozeß soll ein funktionaler Zusammenhang zwischen den mRNA-Konzentrationen eines Gens und den mRNA-Konzentrationen der dieses Gen regulierenden Gene angepaßt werden. Da die mRNA-Konzentration eines Gens aber eigentlich von den Protein-Konzentrationen der es regulierenden Gene abhängig ist, erfordert dieses Verfahren mindestens einen linearen Zusammenhang zwischen den mRNA-Konzentrationen und den Protein-Konzentrationen eines Gens [64]. Dieser ist in der Praxis oftmals nicht gegeben, was die Identifizierung regulatorischer Einflüsse erheblich erschwert.
Ein Vorteil des nichtlinearen Ansatzes ist die Einschränkung der Expressionsrate eines Gens g_i auf das Intervall $(0, max_i)$, was diesen Ansatz gegenüber der linearen Variante biologisch realistischer erscheinen läßt. Diese Beschränkung führt aber auch zu Problemen: Zum einen kann es bei der empirischen Bestimmung der maximalen Expressionsrate max_i passieren, daß alle in den Trainingsdaten beobachteten Expressionsraten von Gen g_i wesentlich kleiner sind als seine maximale Expressionsrate und man deshalb einen zu kleinen Wert für max_i wählt. Zum anderen kann die Expressionsrate eines Gens g_i die Werte 0 und max_i nur asymptotisch annehmen, falls der auf das Gen wirkende regulatorische Einfluß betragsmäßig unendlich groß wird. Treten bei der Arbeit mit realen Expressionsdaten Expressionsraten vom Wert 0 auf, dann müssen Überlegungen getroffen werden, wie diese bei der Anwendung der sigmoidalen Umkehrfunktion (3.33) zu behandeln sind.
Ein wesentlicher Nachteil des Algorithmus selbst ist die Tatsache, daß das Erlernen der Struktur nur implizit über die Schätzung der Gewichtsmatrix erfolgt und zusätzliche Methoden zur Klassifizierung der Schätzer $\hat{w}_{ij}$ notwendig sind.

3.4.2 Evolutionärer Algorithmus

Eine weitere Möglichkeit für das Reverse Engineering im Bereich der Additiven Regulationsmodelle bieten evolutionäre Algorithmen. Die Motivation hierfür liefert die Anwendung dieser Algorithmen im Bereich der neuronalen Netze, zu denen die Additiven Regulationsmodelle große Ähnlichkeit aufweisen. So können die beschriebenen Aktualisierungsregeln (3.28, 3.29) auch zur Definition eines rekurrenten, dynamischen neuronalen Netzwerks dienen. Die Variablen entsprechen dann den Neuronen, die Aktualisierungsregeln der Variablen den Ausgabefunktionen der Neuronen und die Gewichte beschreiben die Verbindungsstärken zwischen den einzelnen Neuronen. Liefert die Anwendung der evolutionären Algorithmen bei dem Erlernen der Architektur eines neuronalen Netzes nur bei sehr kleinen Netzwerken zufriedenstellende Ergebnisse, so können sie doch für das Erlernen der Verbindungsstärken bei bekannter Struktur erfolgreich eingesetzt werden [43, 60]. Diese letztere Problematik ist äquivalent zu der beschriebenen Aufgabenstellung eines Reverse Engineering Algorithmus, der einen guten Schätzer für die Gewichtsmatrix bestimmen muß und dabei zunächst von einer vollständig verknüpften Netzwerkstruktur ausgeht.
Bekannte evolutionäre Algorithmen sind die Evolutionsstrategien und die Genetischen Algorithmen. Sie lösen komplexe Optimierungsprobleme nicht auf dem konventionellen, algorithmischen Weg, sondern nach dem Vorbild der biologischen Evolution und molekularen Genetik. Ziel der Optimierung ist die Minimierung oder Maximierung einer von mehreren Parametern abhängigen Funktion. Aus einer Menge (*Population*) von Parameterzuständen (*Individuen*) wird durch die Anwendung bestimmter, an der Evolution orientierter Operatoren eine neue Generation definierter Parameterzustände erzeugt. Dazu wählt man zunächst aus der Population gemäß dem Darwinschen Prinzip des „survival of the fittest“ die besten Individuen aus (*implizite Selektion - Genetische Algorithmen*). Diese werden dann rekombiniert und mutiert. Die so neu entstandenen Individuen kommen mit einer zu ihrer Fitneß proportionalen Wahrscheinlichkeit in die neue Generation (*explizite Selektion - Evolutionsstrategien*). Nach hinreichend vielen Generationen wird so die optimale Lösung generiert. Detaillierte Erläuterungen sind in [53] zu finden.
Obwohl beide Ansätze – Evolutionsstrategien und Genetische Algorithmen – auf der gleichen Idee basieren, werden die Kodierung der Parameterzustände sowie die Operatoren Rekombination, Mutation und Selektion doch auf verschiedene Weise umgesetzt, wodurch sich ein recht unterschiedliches Verhalten der Algorithmen ergibt. Während die Evolutionsstrategien sehr schnell ein lokales Optimum anstreben und zügig eine weitestgehend homogene Population erreichen, versuchen die Genetischen Algorithmen durch eine schwache Selektion gerade diesen Effekt zu vermeiden. Der für diese Arbeit konstruierte Reverse Engineering Algorithmus ist eher eine Kombination aus beiden Ansätzen. So wurde mit einer, für die Evolutionsstrategien typischen, reell-wertigen Kodierung der Gewichte gearbeitet und entsprechende Rekombinations- und Mutationsoperatoren gewählt, während sich der Selektions-

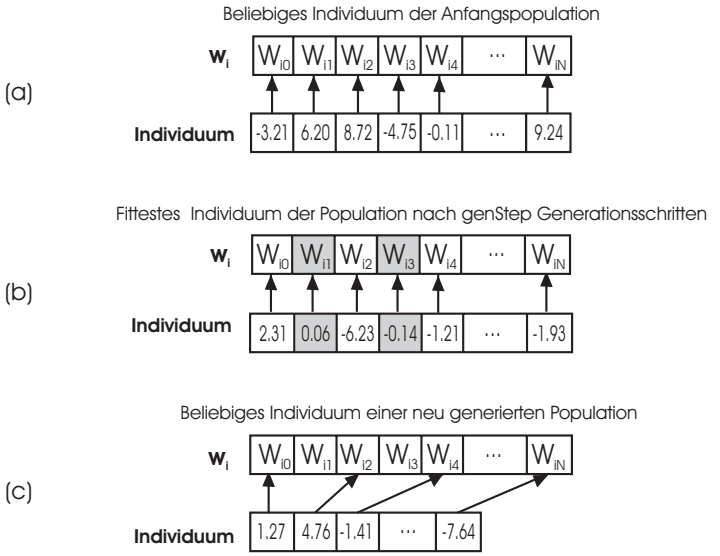


Abbildung 3.5: **Kodierung eines Gewichtsvektors:** (a) Beliebiges Individuum der Anfangspopulation. Jede Position des Individuums enthält den Schätzer eines bestimmten Gewichts. Alle Gewichte des Gewichtsvektors werden kodiert. (b) Fittestes Individuum der Population nach *genStep* Generationsschritten. Der Algorithmus überprüft jetzt jede Position. Ist ein Wert kleiner als der Schwellwert 0.5, kann das zugehörige Gewicht als „null“ klassifiziert und muß im folgenden nicht mehr betrachtet werden. (c) Beliebiges Individuum einer neu generierten Population. Bereits als „null“ klassifizierte Gewichte werden nicht mehr kodiert.

operator an den Genetischen Algorithmen orientiert und eine implizite Selektion benutzt. Einen ähnlichen Ansatz findet man in [5].

Der Algorithmus

Analog zu dem vorangegangenen *REM* Algorithmus müssen auch hier zuerst alle Werte der Outputmatrix Y entsprechend umgeformt werden, wenn der Algorithmus mit einem nichtlinearen Additiven Regulationsmodell arbeitet.
Der Algorithmus generiert für jede Variable x_i den entsprechenden Schätzer $\hat{\mathbf{w}}_i$ ihres Gewichtsvektors in einem separaten Evolutionsprozeß. Die Individuen der jeweiligen Anfangspopulation werden zufällig generiert. Jede Position eines Individuums enthält den Schätzer eines bestimmten Gewichts w_{ij}, und so kodiert jedes Individuum genau einen möglichen Schätzer $\hat{\mathbf{w}}_i$ des wahren Gewichtsvektors $\mathbf{w}_i$ (Abbildung 3.5 (a)). Das Festlegen einer oberen Grenze *maxWeight* und einer unteren Grenze *minWeight* verhindert, daß die Gewichte während des Evolutionsprozesses beliebig groß werden können.
Ausgehend von der Anfangspopulation erzeugt der Algorithmus iterativ neue Generationen der Population. Ein Generationsschritt besteht dabei aus zwei Teilschritten. Im ersten Teilschritt wählt er die k_{elite} fittesten Individuen der aktuellen Generation

– also die „Elite“ – aus und übernimmt sie direkt in die neue Generation. Die Anzahl *popSize* der Individuen in der Population bleibt in jeder Generation konstant, und so werden im zweiten Teilschritt die restlichen *(popSize - k_{elite})* Individuen der neuen Generation produziert. Dafür selektiert der Algorithmus zunächst durch Anwendung des Selektionsoperators aus der aktuellen Generation eine Menge von Individuen, die sich anschließend rekombinieren dürfen. Jedes Individuum der aktuellen Generation wird dabei mit einer zu seiner Fitneß proportionalen Wahrscheinlichkeit ausgewählt und kann auch mehrfach in der selektierten Menge enthalten sein. Der Algorithmus arbeitet hier nach dem bekannten Roulette-Wheel Prinzip:

1. Bewerte jedes Individuum $indiv_k$ mit Hilfe der Fitneßfunktion und ermittle somit seine individuelle Fitneß $fitneß(indiv_k)$.

2. Berechne durch Addition die Fitneß der Population:
   $popFit = \sum_k fitneß(indiv_k)$.

3. Generiere eine Zufallszahl z mit $1 \leq z \leq popFit$.

4. Selektiere ein Individuum $indiv_k$ nach der Berechnung: Finde die kleinste Zahl k mit $\sum\limits_{l \leq k} fitneß(indiv_l) \geq z$.

5. Wiederhole die Schritte 3 und 4, bis *(popSize - k_{elite})* Individuen ausgewählt wurden.

Zur Berechnung der Fitneß eines Individuums werden die gegebenen Trainingsdaten herangezogen. Jedes Individuum $indiv_k$ kodiert genau einen Schätzer $\hat{\mathbf{w}}_{i_k}$ des Gewichtsvektors, und es kann analog zum *REM* Algorithmus der euklidische Fehler berechnet werden. Die Fitneß eines Individuums entspricht dann dem Kehrwert:

$$fitneß(indiv_k) = \frac{1}{\|\mathbf{y}_i - U\hat{\mathbf{w}}_{i_k}^T\|} = \frac{1}{\sum_{m=0}^{M}(y_i^m - \mathbf{u}^m\hat{\mathbf{w}}_{i_k}^T)^2} \tag{3.38}$$

Im Anschluß an die Selektion erzeugt der Algorithmus aus je zwei dieser selektierten Individuen zwei Nachkommen. Mit einer Wahrscheinlichkeit von $1 - p_{recomb}$ werden die beiden Individuen direkt als Nachkommen in die neue Generation aufgenommen; mit einer Wahrscheinlichkeit p_{recomb} werden sie vor der Aufnahme miteinander rekombiniert. Für die Umsetzung des Rekombinationsoperators gibt es prinzipiell mehrere Möglichkeiten. Der hier vorgestellte Algorithmus benutzt den uniformen Überkreuzungsaustausch (*engl.: uniform crossover*): Für jede Position der Individuen entscheidet er zufällig, ob die entsprechenden Werte an dieser Position zwischen den Individuen ausgetauscht werden (Abbildung 3.6).
Abschließend erfolgt noch die Anwendung des Mutationsoperators auf die durch Selektion und Rekombination neu generierten Nachkommen. Jede Position eines jeden dieser Nachkommen wird dabei mit einer Wahrscheinlichkeit p_{mut} durch die

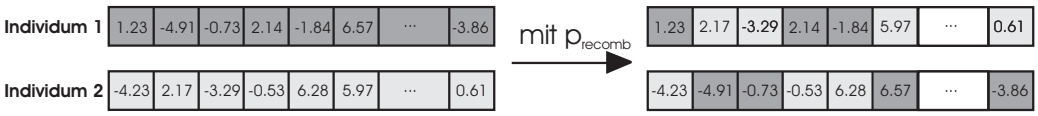


Abbildung 3.6: **Rekombinationsoperator:** Jeweils zwei Individuen dienen zur Generierung von zwei Nachkommen. Dazu werden sie mit einer Wahrscheinlichkeit p_{recomb} miteinander rekombiniert. Für jede ihrer Positionen wird dabei zufällig entschieden, ob die entsprechenden Werte ausgetauscht werden sollen.

Addition einer normalverteilten Zufallsvariable $w_{mut} \sim N(0, \sigma_{mut})$ mutiert. Dabei ist auf die Einhaltung der Grenzen *minWeight* und *maxWeight* zu achten.
Die Klassifikation der einzelnen Gewichte von $\mathbf{w}_i$ in die beiden Klassen „null“ und „nicht-null“ wird vom Algorithmus auf die folgende Weise umgesetzt: Nach einer festgelegten Anzahl *genStep* von Generationsschritten bestimmt er das Individuum mit der höchsten Fitneß und überprüft der Reihe nach alle Positionen dieses Individuums. Ist der Wert einer Position betragsmäßig kleiner als ein vorher festgelegter Schwellwert *threshold*, wird dem zugehörigen Gewicht der Wert 0 zugeordnet und es muß im weiteren Verlauf nicht mehr betrachtet werden. Die Individuen einer anschließend neu generierten Population kodieren nun nur noch die Gewichte des Gewichtsvektors $\mathbf{w}_i$, die noch nicht als „null“ klassifiziert wurden (Abbildung 3.5 (c)). Der Evolutionsprozeß kann nun erneut gestartet werden. Diese Wiederholung des Evolutionsprozesses erfolgt iterativ so oft, bis der Algorithmus bei der nachfolgenden Überprüfung des fittesten Indiviuums keine Positionen mit einem betragsmäßig kleineren Wert als dem Schwellwert finden kann. Alle verbliebenen Gewichte können dann in die Klasse „nicht-null“ eingeordnet werden; ihre im fittesten Individuum kodierten Schätzer bilden dann das Ergebnis der Optimierung.

Zusammenfassend sind hier noch einmal die einzelnen Schritte der iterativen Wiederholung des Evolutionsprozesses aufgelistet:

1. Bestimme eine Anfangspopulation zufällig.
2. Wähle die k_{elite} fittesten Individuen direkt für die folgende Generation aus.
3. Erzeuge die $(popSize - k_{elite})$ restlichen Individuen der folgenden Generation durch die Anwendung der Operatoren Selektion, Rekombination und Mutation.
4. Wiederhole den 2. und 3. Schritt *genStep* mal.
5. Klassifiziere alle Gewichte w_{ij}, für deren Schätzer im fittesten Individuum der Population gilt: $\hat{w}_{ij} < threshold$, als „null“.

6. Falls die Anzahl der in Schritt 5 als „null“ klassifizierten Gewichte größer ist als 0, dann beginne erneut mit Schritt 1. Betrachte dabei nur die noch nicht als „null“ klassifizierten Gewichte.

Implementierung

Für diesen Algorithmus kann der folgende Pseudocode angegeben werden:

```
1   FOR i:=1 TO N
2       find max_i in u_i,y_i
3       FOR m:=1 TO M
4           y_i^m = -ln(max_i / y_i^m - 1)

5   Ŵ : Matrix[N][N+1]
6   FOR i:=1 TO N
7       FOR j:=0 TO N
8           weightIdentifier[j] := j
9       DO
10          population := ∅
11          FOR j:=1 TO popSize
12              indiv_j := ∅
13              FOR k:=0 TO weightIdentifier.length-1
14                  indiv_j[k] := randomValue(minWeight, maxWeight)
15              add indiv_j to population

16          FOR j:=1 TO genStep
17              population:=GENERATION_STEP(population,weightIdentifier)

18          find fittest individuum maxFitIndiv:
19          numberOfNewZeroWeights:=0
20          FOR j:=0 TO maxFitIndiv.length-1
21              IF maxFitIndiv[j] ≤ threshold
22                  delete weightIdentifier[j-numberOfNewZeroWeights]
23                  numberOfNewZeroWeights++
24      WHILE numberOfNewZeroWeights ≥ 0

25      FOR j:=0 TO N
26          Ŵ[i][j] := 0
27      FOR j:=0 TO maxFitIndiv.length-1
28              Ŵ[i][weightIdentifier[j]] := maxFitIndiv[j]

29  PROCEDURE GENERATION_STEP(population,weightIdentifier)
30      COMPUTE_POPFIT(population,weightIdentifier)
31      newGeneration := ∅
32      add the k_elite fittest individuals to newGeneration
33      setOfSelectedIndiv:=SELECTION(population)
34      newGeneration:=RECOMBINATION(setOfSelectedIndiv,newGeneration)
```

```
35          newGeneration:=MUTATION(newGeneration)
36          RETURN newGeneration

37      PROCEDURE COMPUTE_POPFIT(population,weightIdentifier)
38          population.fitness:=0
39          FOR j:=1 TO popSize
40              population+=COMPUTE_INDIVFIT(j,weightIdentifier)

41      PROCEDURE COMPUTE_INDIVFIT(j,weightIdentifier)
42          e := 0
43          FOR m:=1 TO M
44              ŷ_i^m := 0
45              FOR k:=0 TO weightIdentifier.length-1
46                  ŷ_i^m += u^m_{weightIdentifier[k]} · indiv_j[k]
47              e+ = (y_i^m − ŷ_i^m)^2
48          fitness(indiv_j) = 1/e
49          RETURN fitness(indiv_i)

50      PROCEDURE SELECTION(population)
51          set := ∅
52          select (popSize − k_elite) individuals using Roulette-Wheel; put them into the set
53          RETURN set

54      PROCEDURE RECOMBINATION(set,newGeneration)
55          FOR k:=1 TO set.length/2
56              IF randomValue(0,1) > p_recomb
57                  add set[2·k−1], set[2·k] to newGeneration
58              ELSE
59                  FOR l:=0 TO set[2·k].length-1
60                      IF randomValue(0,1) > 0.5
61                          exchange set[2·k][l] and set[2·k−1][l]
62                  add set[2·k−1], set[2·k] to newGeneration
63          RETURN newGeneration

64      PROCEDURE MUTATION(newGeneration)
65          FOR k:=k_elite TO newGeneration.length
66              FOR l:=0 TO newGeneration[k].length-1
67                  IF randomValue(0,1) < p_mut
68                      newGeneration[k][l]+ = N(0, σ_mut)
69          RETURN newGeneration
```

Im ersten Teil (Zeilen 1-4) werden analog zum *REM* Algorithmus die maximalen Expressionsraten bestimmt und die entsprechende Umformung der Werte der Outputmatrix vorgenommen. Er ist nur zu implementieren, wenn man mit einem nichtlinearen Additiven Regulationsmodell arbeitet.

Den Schätzer $\hat{\mathbf{w}}_i$ des wahren Gewichtsvektors $\mathbf{w}_i$ bestimmt der Algorithmus durch die iterative Wiederholung eines Evolutionsprozesses (Zeilen 9-24). In jeder Wiederholung muß dabei zuerst die entsprechende Anfangspopulation zufällig generiert werden (Zeilen 10-15). Ein Individuum kodiert jeweils alle noch nicht als „null“ klassifizierten Gewichte. Für die Zuordnung der einzelnen Positionen der Individuen zu den entsprechenden Gewichten dient der Vektor *weightIdentifier*, indem er, im Unterschied zu den Individuen, nicht den Wert eines Gewichtes w_{ij}, sondern ihren entsprechenden Index j enthält.
Iterativ werden durch Anwendung der Prozedur *GENERATION_STEP* (Zeilen 29-36), die einen Generationsschritt implementiert, neue Generationen der Population erzeugt (Zeilen 16-17). Die dabei verwendeten Prozeduren *COMPURE_POPFIT* (Zeilen 37-40), *COMPUTE_INDIVFIT* (Zeilen 41-49), *SELECTION* (Zeilen 50-53), *RECOMBINATION* (Zeilen 54-63) und *MUTATION* (Zeilen 64-69) implementieren die im vorangegangenen Unterabschnitt beschriebenen Techniken. Nach *genStep* Generationsschritten werden die Werte aller Positionen des fittesten Individuum überprüft (Zeilen 20-23). Ein betragsmäßig kleinerer Wert als *threshold* klassifiziert das zugehörige Gewicht als „null“. Der entsprechende Index dieses Gewichtes kann aus dem Vektor *weightIdentifier* gelöscht werden (Zeile 22), da die Individuen dieses Gewicht bei der nächsten Wiederholung des Evolutionsprozesses nicht mehr kodieren müssen. Wird bei dieser Überprüfung kein zusätzliches Gewicht in die Klasse „null“ eingeordnet, bricht der Algorithmus die iterative Wiederholung des Evolutionsprozesses ab (Zeile 24) und trägt die Werte des fittesten Individuums in die Gewichtsmatrix $\hat{W}$ ein (Zeilen 25-28).

Limitationen

Grundsätzlich gelten für diesen Algorithmus die gleichen Limitationen, die bereits im vorangegangenen Abschnitt 3.4.1 für den *REM* Algorithmus beschrieben wurden. Zusätzlich erschwerend kommt hinzu, daß die Ergebnisse stark von der Wahl der Parameter des Algorithmus abhängig sind. Prinzipiell wirken sich eine größere Population und eine höhere Anzahl an Generationsschritten positiv auf die Ergebnisse aus, verschlechtern andererseits aber die Laufzeit des Algorithmus erheblich. Auch für die Rekombinationswahrscheinlichkeit p_{recomb} und die Mutationswahrscheinlichkeit p_{mut} existieren keine problemübergreifenden optimalen Werte. Sie müssen genau wie der Schwellwert $threshold$ und die Größe k_{elite} der Elite in Testläufen sorgfältig bestimmt werden. Bei den in dieser Arbeit durchgeführten Simulationsexperimenten wurde mit einer Populationsgröße $popSize := 500$ gearbeitet; die Anzahl der Generationsschritte *genStep* betrug jeweils 50. Untersuchungen ergaben, daß mit $p_{recomb} := 1$, $p_{mut} := 0.075$, $threshold := 0.5$ und $k_{elite} = 10$ die besten Ergebnisse erzielt werden konnten. Inwiefern sich diese speziellen Werte auf die Anwendung des Algorithmus mit realen Expressionsdaten übertragen lassen, bleibt offen.

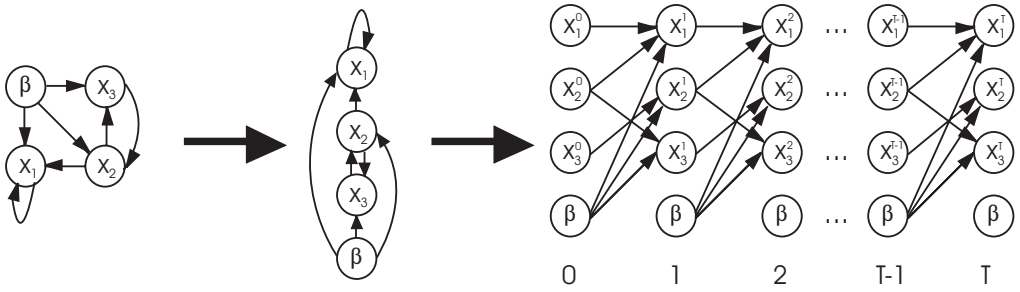


Abbildung 3.7: **Transformation eines rekurrenten Netzwerks in ein Feedforward-Netzwerk [13].**

3.4.3 BPTT - Backpropagation through time (D'haeseleer [13])

Wie im vorherigen Abschnitt 3.4.2 beschrieben, ist es möglich, die Aktualisierungsregeln des Additiven Regulationsmodells auch als Definition eines dynamischen, rekurrenten, neuronalen Netzwerks aufzufassen. Daraus ergibt sich der Vorteil, daß man auf effiziente Algorithmen, die im Bereich der neuronalen Netzwerke zur Anpassung der Verbindungsstärken entwickelt worden sind, zurückgreifen kann. An dieser Stelle soll deshalb ein Ansatz von D'haeseleer [13] vorgestellt werden, der einen Lernalgorithmus für Feedforward-Netzwerkarchitekturen – den Backpropagation through time Algorithmus – für das Reverse Engineering in Additiven Regulationsmodellen einsetzt.
Im Gegensatz zu den vorangegangenen zwei Algorithmen, die sowohl mit Zeitreihen als auch mit Zustandsübergangsdaten arbeiten können, erwartet dieser Algorithmus als Eingabe explizit eine Zeitreihe, die den Systemzustand des Genregulationsnetzwerks an $T+1$ aufeinanderfolgenden Zeitpunkten beschreibt:

$$D = \begin{pmatrix} d^0 \\ \vdots \\ d^T \end{pmatrix} = \begin{pmatrix} 1 & d_1^0 & \cdots & d_N^0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & d_1^T & \cdots & d_N^T \end{pmatrix} \tag{3.39}$$

Aufgrund der Integration der Biasfaktoren in die Gewichtsmatrix kann man die Biasfaktoren als eine zusätzliche Variable im Netzwerk mit einem konstanten Wert von 1 auffassen. Dieser konstante Wert wird durch den Spaltenvektor d_0 modelliert. Für die Anwendung des BPTT Algorithmus muß das gegebene, rekurrente Netzwerk zunächst in ein Feedforward-Netzwerk transformiert werden. Beginnend mit einer Schicht zum Zeitpunkt $t = 0$, ist schrittweise mit jedem weiteren gegebenen Zeitpunkt eine Schicht mit identischer Gewichtsmatrix hinzuzufügen (Abbildung 3.7). Auf diese Weise läßt sich eine zeitliche Entkopplung erreichen; der Zeitparameter t entspricht dann nur noch einem Parameter der Schichtenanzahl.

Die Grundidee des BPTT Lernalgorithmus ist die iterative Minimierung einer von den Gewichten abhängigen Fehlerfunktion E durch ein Gradientenabstiegsverfahren. Beginnend mit zufällig gewählten Belegungen für die einzelnen Gewichte, müssen diese in jeder Iteration jeweils in die Richtung des minimalsten Fehlers aktualisiert werden. In der Regel entspricht E dem quadratischen Fehler zwischen den in den Trainingsdaten beobachteten Werten d_i^t und den vom Netzwerk mit Hilfe der aktuellen Gewichte vorhergesagten Werten x_i^t:

$$E = \frac{1}{2} \sum_{t=0}^{T} \sum_{i=1}^{N} (d_i^t - x_i^t)^2 \tag{3.40}$$

Die Aktualisierungsregel für ein Gewicht w_{ij} ergibt sich aus der partiellen Ableitung der Fehlerfunktion E nach eben diesem Gewicht multipliziert mit der Lernrate η:

$$\Delta w_{ij} = -\eta \cdot \frac{\delta E}{\delta w_{ij}} \tag{3.41}$$

Prinzipiell kann dieser Algorithmus sowohl auf ein lineares als auch auf ein nichtlineares Additives Regulationsmodell angewandt werden. In der Praxis führt die Anwendung auf das lineare Modell allerdings nur selten zu akzeptablen Ergebnissen. Der Vollständigkeit wegen soll die lineare Variante hier trotzdem betrachtet werden.

Der Algorithmus

Wird mit einem nichtlinearen Additiven Regulationsmodell gearbeitet, besteht auch bei diesem Ansatz die erste Aufgabe des Algorithmus darin, die maximalen Expressionsraten aus den gegebenen Trainingsdaten D zu schätzen. Sie werden für spätere Berechnungen benötigt; eine Umformung der gegebenen Expressionsraten nach 3.33 (sigmoidale Umkehrfunktion) ist hier allerdings nicht erforderlich.
Der Algorithmus startet mit einer zufälligen Initialisierung der Gewichte. Durch die iterative Ausführung eines aus zwei Phasen bestehenden Lernschrittes werden die Gewichte anschließend so angepaßt, daß sie die Fehlerfunktion E minimieren.
In der ersten Phase – der Feedforward-Berechnung – speist der Algorithmus den Eingabevektor d^0 in das Netzwerk ein und propagiert ihn durch das Netzwerk hindurch:

linearer Ansatz:

$$x_i^0 = d_i^0 \qquad \text{und} \qquad x_i^t = \sum_{j=0}^{N} w_{ij} x_j^{t-1} = r_i^t \tag{3.42}$$

nichtlinearer Ansatz:

$$x_i^0 = d_i^0 \qquad \text{und} \qquad x_i^t = S(\sum_{j=0}^{N} w_{ij} x_j^{t-1}) = S(r_i^t) \tag{3.43}$$

Die zweite Phase – die Backward-Berechnung – dient dann zur Aktualisierung der Gewichte. Um die Änderungen Δw_{ij} zu bestimmen, greift der Algorithmus auf die Kettenregel

$$\frac{\delta E(r_i(w_{ij}^t))}{\delta w_{ij}^t} = \frac{\delta E(r_i(w_{ij}^t))}{\delta r_i(w_{ij})} \cdot \frac{\delta r_i(w_{ij})}{\delta w_{ij}^t} \tag{3.44}$$

zurück. Rekursiv berechnet er zunächst für jeden Knoten des Netzwerks den entsprechenden Fehlergradienten $\delta_i(t) = \frac{\delta E}{\delta r_i^t}$. Beginnend bei der Schicht T wird der Fehler so durch das Netzwerk zurück propagiert. Sei $\delta_i^s(t)$ dabei der Anteil des Fehlergradienten $\delta_i(t)$, der am Knoten x_i^t durch Netzwerkaktivitäten in den nachfolgenden Schichten entsteht:

$$\delta_i^s(t) = \begin{cases} 0 & t = T \\ -\frac{\delta x_i^t}{\delta r_i^t} \sum_{j=1}^{N} w_{ji} \delta_j(t+1) & 0 < t < T \end{cases} \tag{3.45}$$

Analog beschreibt $\delta_i^z(t)$ den Anteil des Fehlergradienten $\delta_i(t)$, der am Knoten x_i^t selbst hervorgerufen wird:

$$\delta_i^s(t) = -\frac{\delta x_i^t}{\delta r_i^t}(d_i^t - x_i^t) \tag{3.46}$$

Aus der Addition dieser beiden Gradienten $\delta_i^s(t)$ und $\delta_i^z(t)$ resultiert dann der Gesamtfehlergradient $\delta_i(t)$:

$$\delta_i(t) = \begin{cases} -\frac{\delta x_i^t}{\delta r_i^t}(d_i^t - x_i^t) & t = T \\ -\frac{\delta x_i^t}{\delta r_i^t} \cdot ((d_i^t - x_i^t) + \sum_{j=1}^{N} w_{ji} \delta_j(t+1)) & 0 < t < T \end{cases} \tag{3.47}$$

mit:

$$\frac{\delta x_i^t}{\delta r_i^t} = \frac{\delta r_i^t}{\delta r_i^t} = 1 \qquad \text{(linearer Ansatz)}$$

$$\frac{\delta x_i^t}{\delta r_i^t} = \frac{\delta S(r_i^t)}{\delta r_i^t} = S(r_i^t)(1 - S(r_i^t)) = x_i^t(1 - x_i^t) \qquad \text{(nichtlinearer Ansatz)}$$

Mit Hilfe dieser berechneten Fehlergradienten können anschließend die Gewichte des Netzwerks entsprechend aktualisiert werden.

$$w_{ij}^{neu} = w_{ij}^{alt} + \Delta w_{ij}$$

mit:

$$\begin{aligned}
\Delta w_{ij} &= -\eta \frac{\delta E}{\delta w_{ij}} \\
&= -\eta \sum_{t=1}^{T} \frac{\delta E}{\delta w_{ij}^t} \\
&= -\eta \sum_{t=1}^{T} \frac{\delta E}{\delta r_i^t} \cdot \frac{\delta r_i^t}{\delta w_{ij}^t} \\
&= -\eta \sum_{t=1}^{T} \delta_i(t) \cdot x_j^{t-1} \qquad (3.48)
\end{aligned}$$

Normalerweise werden diese zwei Phasen iterativ so oft wiederholt, bis der Fehler E des Netzwerks kleiner als ein vorgegebener Schwellwert ist. Das Netzwerkmodell kann so beliebig gut an die gegebenen Trainingsdaten angepaßt werden. Das führt leicht zu einer Überanpassung (*engl.: overfitting*) der Gewichte und einer daraus resultierenden Einschränkung der Generalisierbarkeit des Netzwerkmodells. Um diese Probleme zu vermeiden, wird in [13] vorgeschlagen, nach einer gewissen Anzahl an Lernschritten das Training unabhängig vom verbleibenden Fehler E zu beenden.
Die Klassifizierung der Gewichte in die beiden Gruppen „null“ und „nicht-null“ geschieht bei diesem Algorithmus im Anschluß an das Training. Dazu wird die Verteilung der aus dem Training hervorgegangenen Gewichte durch eine aus den zwei Normalverteilungen $N_{zero}(0, \sigma_{zero})$ und $N_{nonzero}(0, \sigma_{nonzero})$ zusammengesetzte Mischverteilung approximiert. N_{zero} und $N_{nonzero}$ beschreiben dabei die Verteilungen der Gewichte in den Klassen „null“ und „nicht-null“. Die Dichtefunktion der Mischverteilung ergibt sich daraus wie folgt:

$$f(w) = p_{zero} \cdot \frac{1}{\sqrt{2\Pi}\sigma_{zero}} \cdot e^{-\frac{w^2}{2\sigma_{zero}^2}} + p_{nonzero} \cdot \frac{1}{\sqrt{2\Pi}\sigma_{nonzero}} \cdot e^{-\frac{w^2}{2\sigma_{nonzero}^2}} \qquad (3.49)$$

Dabei gibt der Parameter p_{zero} die Wahrscheinlichkeit an, daß ein Gewicht zur Klasse „null“ gehört. Analog ist der Parameter $p_{nonzero} = (1 - p_{zero})$ zu interpretieren. Mit Hilfe der Maximum Likelihood Schätzung lassen sich entsprechende Schätzer $\hat{p}_{zero}$, $\hat{\sigma}_{zero}$ und $\hat{\sigma}_{nonzero}$ für die Parameter der Mischverteilung finden. Allerdings ist dieses Maximierungsproblem analytisch nicht lösbar, und es wurde hier zur numerischen Berechnung der Schätzer auf das Statistikprogramm R [58] zurückgegriffen. Mit Hilfe der Schätzer $\hat{\sigma}_{zero}$ und $\hat{\sigma}_{nonzero}$ kann ein Gewicht w_{ij} anschließend genau dann in die Klasse „nicht-null“ eingeordnet werden, falls sein Wert in dieser Klasse wahrscheinlicher ist. Man überprüft dazu die Ungleichung $f_{zero}(w_{ij}) < f_{nonzero}(w_{ij})$.

Prinzipiell sind auch andere Methoden zur Klassifizierung der Gewichte möglich. Analog zum vorangegangenen *evolutionären Algorithmus* könnte man zum Beispiel schon während des Trainings betragsmäßig kleine Gewichte als „null“ klassifizieren und ihren Wert auf 0 festlegen.

Implementierung

Für die Implementierung des Algorithmus lautet der Pseudocode unter der Annahme, daß zur Modellierung des Genregulationsnetzwerks ein nichtlineares Additives Regulationsmodell verwendet wird, wie folgt:

1 **FOR** *i:=1* **TO** N
2 *find* max_i *in* d_i

3 $\hat{W} : Matrix[N][N+1]$
4 **FOR** *i:=1* **TO** N
5 **FOR** *j:=0* **TO** N
6 $\hat{W}[i][j] := randomValue(-1, 1)$

7 **FOR** *k:=1* **TO** *learningStep*
8 X:=FORWARD_PASS($\hat{W}$)
9 $\hat{W}$:=BACKWARD_PASS(X,$\hat{W}$)

10 *estimate the distribution parameters* p_{zero}, σ_{zero} *and* $\sigma_{nonzero}$ of the weights in $\hat{W}$
11 **FOR** *i:=1* **TO** N
12 **FOR** *j:=0* **TO** N
13 **IF** $\left(\frac{1}{\sigma_{zero}} \cdot e^{-\frac{\hat{W}[i][j]^2}{2\sigma^2_{zero}}}\right) > \left(\frac{1}{\sigma_{nonzero}} \cdot e^{-\frac{\hat{W}[i][j]^2}{2\sigma^2_{nonzero}}}\right)$
14 $\hat{W}[i][j] := 0$)

15 **PROCEDURE** FORWARD_PASS($\hat{W}$)
16 $X : Matrix[T+1][N+1]$
17 $X[0] := d^0$
18 **FOR** *t:=1* **TO** T
19 $X[t][0] := 1$
20 **FOR** *i:=1* **TO** N
21 $X[t][i] := S(X[t-1]\hat{W}[i]^T)$
22 **RETURN** X

23 **PROCEDURE** BACKWARD_PASS(X,$\hat{W}$)
24 G:=COMPUTE_GRADIENTS(X,$\hat{W}$)
25 **FOR** *i:=1* **TO** N
26 **FOR** *j:=0* **TO** N
27 $\hat{W}[i][j] := -\eta \sum_{t=1}^{T} G[t][i] \cdot X[t-1][j]$
28 **RETURN** $\hat{W}$

```
29  PROCEDURE COMPUTE_GRADIENT(X,Ŵ)
30      G : Matrix[T][N]
31      FOR i:=1 TO N
32          G[T][i] := (X[T][i] − d_i^T)X[T][i](1 − X[T][i])
33      FOR t:=T-1 DOWNTO 1
34          FOR i:=1 TO N
35              G[t][i] := ((X[t][i] − d_i^T) + Σ_{j=1}^{N} Ŵ[i][j] · G[t + 1][j])
36                      ·X[t][i](1 − X[t][i])
37      RETURN G
```

Nachdem im ersten Teil (Zeilen 1-2) die maximalen Expressionsraten aus den Trainingsdaten identifiziert wurden, erfolgt im zweiten Teil (Zeilen 3-6) eine zufällige Initialisierung der Gewichte. Dazu werden gemäß der Annahme, daß der wahre Wert vieler Gewichte 0 ist, nur kleine Zufallszahlen erzeugt.
Anschließend beginnt der Algorithmus im dritten Teil (Zeilen 7-9) mit der iterativen Wiederholung des Lernschrittes. Die erste Phase – die Forward-Berechnung – ist dabei in der Prozedur *FORWARD_PASS* implementiert (Zeilen 15-22). Die Prozedur *BACKWARD_PASS* (Zeilen 23-28) setzt dagegen die zweite Phase des Lernschrittes um. Nach der rekursiven Berechnung der Fehlergradienten mit Hilfe der Prozedur *COMPUTE_GRADIENT* (Zeilen 29-37) werden diese für die Aktualisierung der Gewichte verwendet. Das Training endet nach *learningStep* Iterationen des Lernschrittes. Zur Klassifikation der Gewichte müssen dann die entsprechenden Parameter der Verteilung der aus dem Training hervorgegangenen Gewichte geschätzt werden (Zeile 10). Wie schon erwähnt, wurde hierbei auf das Statistikprogramm R zurückgegriffen. Ist das Auftreten eines Gewichts in der Klasse „null“ wahrscheinlicher als in der Klasse „nicht-null“, wird der zugehörige Wert des Gewichts in der Gewichtsmatrix $\hat{W}$ auf 0 gesetzt (Zeilen 13-14).

Limitationen

An dieser Stelle kann wieder auf den Unterabschnitt *Limitationen* des *REM* Algorithmus verwiesen werden. Zusätzlich erschwerend kommen hier Probleme bei der Wahl der Lernrate η hinzu, die einen erheblichen Einfluß auf die Laufzeit des Algorithmus ausübt. Die Lernrate ist problemabhängig, und es kann kein allgemeingültiger, optimaler Wert angegeben werden. Bei der Implementierung des Algorithmus in dieser Arbeit wurde eine konstante Lernrate verwendet. Experimentelle Untersuchungen ergaben, daß der Algorithmus bei einer Lernrate von $\eta = 0.35$ die besten Ergebnisse liefert. Prinzipiell hat eine große Lernrate auch große Sprünge in der Fehlerlandschaft zur Folge. Der Algorithmus strebt dann zwar sehr schnell ein Minimum der Fehlerfunktion E an, es können aber als Folge der großen Sprünge auch Oszillationen enstehen, bei denen der Algorithmus immer wieder über das Minimum hinaus springt. Ist die Lernrate hingegen klein, werden sehr viele Iterationen

benötigt, denn die Veränderungen der Gewichte sind dann ebenfalls klein und der Algorithmus strebt nur sehr langsam auf ein Minimum von E zu. Abhilfe schaffen kann die Verwendung einer abnehmenden Lernrate. Als Richtlinie gilt dabei: $\eta_{akt} = \frac{1}{\#Iterationsschrittes}$. Ferner ist die Lernrate auch von der lokalen Fehlerlandschaft des aktuellen Lernschrittes abhängig und für jede Iteration des Lernschrittes kann es eine andere optimale Lernrate geben; dasselbe gilt auch für die einzelnen Gewichte w_{ij}. Die beste Strategie zur Lösung dieser Problematik bietet eine iterative Anpassung der Lernrate an die jeweilige lokale Fehlerlandschaft. Eine Möglichkeit bietet hier beispielsweise der Delta-bar-Delta [30] Algorithmus. Jedes Gewicht w_{ij} erhält dabei eine eigene Lernrate η_{ij}, die bei jeder Iteration entsprechend des Fehlergradienten $\frac{\delta E}{\delta w_{ij}}$ aktualisiert wird. Grob kann die hierbei verfolgte Strategie so beschrieben werden: Zeigt der Fehlergradient von w_{ij} im aktuellen Lernschritt in die gleiche Richtung wie der entsprechende, durchschnittliche Gradient der vorangegangenen Lernschritte, wird die Lernrate η_{ij} vergrößert; ansonsten verkleinert. Der Delta-bar-Delta Algorithmus kann zwar etwas Verbesserung bringen, ist aber seinerseits ebenfalls parameterabhängig. Die Wahl der Belegung dieser zusätzlichen Parameter ist nicht trivial.
Weiterhin sind die Ergebnisse von der Anzahl *learningStep* der Iterationen eines Lernschrittes abhängig. Wie erwähnt, kann das Modell durch eine größere Anzahl an Iterationen besser an die Trainingsdaten angepaßt werden, bringt aber auch das Problem der Überanpassung mit sich. In [13] wurde, ebenfalls durch experimentelle Untersuchungen, eine Anzahl von 2000 Lernschritten als optimales Abbruchkriterium bestimmt. Wieder bleibt offen, inwieweit sich diese Werte für η und *learningStep* auf die Arbeit mit realen Genexpressionsdaten übertragen lassen.

3.5 Reverse Engineering in kontinuierlichen dynamischen Bayesschen Netzen

In einem kontinuierlichen DBN $B = \langle G, \Theta_G \rangle$ wird die Expressionsrate eines Gens g_i zu einem Zeitpunkt t durch eine kontinuierliche Zufallsvariable $X_i[t]$ beschrieben. Die bedingte Wahrscheinlichkeitsverteilung einer Zufallsvariablen $X_i[t]$ kann in Anlehnung an das Additive Regulationsmodell durch eine Normalverteilung modelliert werden, die alle prinzipiell möglichen regulatorischen Einflüsse der Zufallsvariablen in $\mathbf{X}[t]$, also der Expressionsraten aller Gene zum vorangegangenen Zeitpunkt, berücksichtigt:

linearer Ansatz:

$$X_i[t+1] \sim N(\sum_{j=0}^{N} w_{ij} X_j[t], \sigma^2) \tag{3.50}$$

nichtlinearer Ansatz:

$$X_i[t+1] \sim N(S(\sum_{j=0}^{N} w_{ij} X_j[t]), \sigma^2) \tag{3.51}$$

Wie zu erkennen, können auch hier durch das Einführen einer zusätzlichen Variablen $X_0[t]$ die Biasfaktoren als Spaltenvektor in die Gewichtsmatrix W integriert werden. Die Anlehnung an das Additive Regulationsmodell impliziert außerdem die vereinfachte Betrachtung der Genregulationsprozesse als stationäre Markov-Prozesse.
Analog zu den diskreten DBN besteht die Aufgabe eines Reverse Engineering Algorithmus zum einen darin, einen möglichst guten Schätzer $\hat{G}$ der wahren Struktur zu finden, der angibt, wie die Zufallsvariablen in $\mathbf{X}[t+1]$ von den Zufallsvariablen in $\mathbf{X}[t]$ abhängen. Zum anderen müssen die bedingten Wahrscheinlichkeitsverteilungen der Zufallsvariablen in $\mathbf{X}[t+1]$ spezifiziert werden. Letzteres geschieht hier im Gegensatz zu den diskreten DBN, bei denen die einzelnen Wahrscheinlichkeiten θ_{i,x_i,pa_i} geschätzt werden müssen, durch das Schätzen der Parameter w_{ij} der bedingten Wahrscheinlichkeitsverteilungen.
Die aus M Trainingsvektoren bestehenden Trainingsdaten $D = \{d_1, d_2, ..., d_M\}$ mit

$$d_m = \begin{pmatrix} x_{1,m}[t] & x_{1,m}[t+1] \\ \vdots & \vdots \\ x_{N,m}[t] & x_{N,m}[t+1] \end{pmatrix} \tag{3.52}$$

entsprechen auch hier wahlweise Zustandsübergangsdaten oder einer Zeitreihe (siehe Abschnitt 3.3.1). Analog zum Additiven Regulationsmodell können diese Daten auch in Form einer Inputmatrix U und einer Outputmatrix Y aufbereitet werden (siehe Abschnitt 3.4). Der m-te Zeilenvektor von U beschreibt dann die Zustände der Variablen $\mathbf{X}[t]$ in dem Trainingsvektor d_m; der m-te Zeilenvektor von Y enthält die entsprechenden Zustände der Variablen $\mathbf{X}[t+1]$ in d_m. Der zusätzliche Spaltenvektor $\mathbf{u}_0$ modelliert den konstanten Zustand der für die Biasfaktoren eingeführten Zufallsvariablen $X_0[t]$. Arbeitet man mit dem nichtlinearen Ansatz, empfiehlt es sich auch hier, die maximalen Expressionsraten zu bestimmen, die in der Outputmatrix Y gegebenen Werte der Zufallsvariablen $\mathbf{X}[t+1]$ nach 3.33 umzuformen und so die Zusammenhänge zwischen den Zufallsvariablen $\mathbf{X}[t+1]$ und ihren jeweiligen Eltern aus $\mathbf{X}[t]$ zu linearisieren.
Die Struktur G eines kontinuierlichen DBN wird in diesem speziellen Ansatz vollständig durch die in Θ_G enthaltene Gewichtsmatrix W festgelegt, denn ein Wert ungleich 0 eines Gewichtes w_{ij} gibt an, daß der mögliche regulatorische Einfluß von $X_j[t]$ auf $X_i[t+1]$ tatsächlich vorhanden ist, während ein Wert von 0 dafür steht, daß $X_i[t+1]$ nicht von $X_j[t]$ abhängt. Aufgrund dieser Tatsache argumentiert man in [46], daß analog zu den Additiven Regulationsmodellen beim Schätzen der Parameter w_{ij} implizit die Struktur erlernt werden kann und sich so das Lernen der Struktur bei

diesem Ansatz auf das Schätzen der Parametervektoren $\mathbf{w}_i$ reduzieren läßt. Man geht also wiederum von einer vollständig verknüpften Struktur G aus und berechnet den zugehörigen Maximum Likelihood Schätzer $\hat{\Theta}_G$ der Parametermenge Θ_G. Wie in Abschnitt 3.3 beschrieben, muß dafür die Likelihoodfunktion

$$L(\Theta_G : D|G) = P(D|\langle G, \Theta_G \rangle), \tag{3.53}$$

die die Wahrscheinlichkeit der Daten in Abhängigkeit von Θ_G beschreibt, bezüglich der einzelnen Vektoren $\mathbf{w}_i$ maximiert werden. Es ist möglich zu zeigen, daß die Maximum Likelihood Schätzung eines Gewichtsvektors $\mathbf{w}_i$, die aus dem Maximieren der Likelihoodfunktion bezüglich $\mathbf{w}_i$ resultiert, der Schätzung von $\mathbf{w}_i$ durch die Methode der kleinsten Quadrate entspricht [46] (siehe Anhang A). Damit ist dieser, in [46] vorgeschlagene Reverse Engineering Ansatz äquivalent zum Reverse Engineering in Additiven Regulationsmodellen. Dies verdeutlicht den folgenden Nachteil der Herangehensweise in [46]: Fehlerbehaftete Trainingsdaten erschweren das implizite Erlernen der Struktur erheblich. Denn auch wenn für das wahre Gewicht gilt: $w_{ij} = 0$, und somit kein regulatorischer Einfluß von $X_j[t]$ auf $X_i[t+1]$ ausgeht, wird der entsprechende Schätzer $\hat{w}_{ij}$ aufgrund von Fehlern und Inkonsistenzen in den Daten vielleicht zwar einen kleinen Wert annehmen, aber nur sehr selten direkt dem Wert 0 entsprechen. Es müssen deshalb zusätzliche Methoden herangezogen werden, die die geschätzten Parameter in „null“ und „nicht-null“ klassifizieren (vergleiche Abschnitt 3.4).
Da effiziente Algorithmen zum Erlernen der Struktur von diskreten DBN existieren, soll diese Arbeit untersuchen, ob der dafür in Abschnitt 3.3.1 beschriebene Algorithmus auch für kontinuierliche DBN effizient eingesetzt und damit das Reverse Engineering – im Gegensatz zu der Herangehensweise in [46] – durch ein explizites Erlernen der Struktur unterstützt werden kann.

3.5.1 Lernalgorithmus zur Identifizierung der Struktur eines kontinuierlichen DBN

Zur Erinnerung läßt sich die Grundidee des Lernalgorithmus für diskrete DBN wie folgt kurz zusammenfassen: Die Güte einer Struktur wird durch die BIC-Scoring-Funktion festgelegt. Ausgehend von einer Startstruktur beginnt der Algorithmus eine heuristische Suche durch den Suchraum möglicher Strukturen und versucht durch das iterative Löschen oder Einfügen einer Kante, eine Struktur mit einem höheren BIC-Score zu finden.

Der Algorithmus

Eine detaillierte Beschreibung des Algorithmus ist in Abschnitt 3.3.1 nachzulesen. Für die Anwendung des Algorithmus auf kontinuierliche DBN mußte lediglich eine

Anpassung des BIC-Scores

$$Score_{BIC}(G, D) = \log L(\langle G, \hat{\Theta}_G \rangle : D) - \frac{\log M}{2} dimG \tag{3.54}$$

an das kontinuierliche Modell erfolgen. Die Umformungen der Likelihood Funktion sind in Anhang A detailliert dargestellt. Damit ergibt sich aus 3.54:

$$\begin{aligned} & \sum_{i=1}^{N} \left(\log \frac{1}{(2\pi)^{\frac{M}{2}} \sigma^M} - \sum_{m=1}^{M} \frac{(y_i^m - \mathbf{w}_i \mathbf{u}^{m^T})^2}{2\sigma^2} \right) - \frac{\log M}{2} dimG \\ = \quad & \log \frac{N}{(2\pi)^{\frac{M}{2}} \sigma^M} - \sum_{i=1}^{N} \sum_{m=1}^{M} \frac{(y_i^m - \hat{\mathbf{w}}_i \mathbf{u}^{m^T})^2}{2\sigma^2} - \frac{\log M}{2\sigma^2} dimG \cdot \sigma^2 \end{aligned} \tag{3.55}$$

Der erste Term ist eine additive, strukturunabhängige Konstante und muß nicht weiter berücksichtigt werden:

$$Score_{BIC}(G, D) \propto - \sum_{i=1}^{N} \sum_{m=1}^{M} \frac{(y_i^m - \hat{\mathbf{w}}_i \mathbf{u}^{m^T})^2}{2\sigma^2} - \frac{\log M}{2\sigma^2} dimG \cdot \sigma^2 \tag{3.56}$$

Aus dem gleichen Grund läßt sich auch der konstante Faktor $\frac{1}{2\sigma^2}$ streichen:

$$Score_{BIC}(G, D) \propto - \sum_{i=1}^{N} \sum_{m=1}^{M} (y_i^m - \hat{\mathbf{w}}_i \mathbf{u}^{m^T})^2 - \log M \cdot dimG \cdot \sigma^2 \tag{3.57}$$

Wie zu erkennen, entspricht der erste Term der Summe der Fehlerquadrate zwischen den Einträgen der beobachteten Outputmatrix Y und den vom Netzwerk $B = \langle G, \hat{\Theta}_G \rangle$ vorhergesagten Werten. Aufgrund der in Anhang A gezeigten Äquivalenz zwischen der Maximum Likelihood Schätzung und der Methode der kleinsten Quadrate, kann man zur Berechnung der Maximum Likelihood Schätzer $\hat{\mathbf{w}}_i$ die Methode der kleinsten Quadrate heranziehen. Dabei müssen nur diejenigen Gewichte w_{ij} betrachtet werden, deren zugehörige regulatorische Einflüsse in der Struktur G enthalten sind. Gewichte, deren entsprechenden Kanten in G nicht existieren, können vorher auf den Wert 0 festgelegt werden.
Der Strafterm berücksichtigt analog zum diskreten Ansatz die Komplexität der Struktur. Er ist hier außerdem zusätzlich von der Varianz σ^2 abhängig, was wie folgt interpretiert werden kann: Die Varianz σ^2 ist ein Parameter zur Beschreibung der Stochastizität der regulatorischen Einflüsse. Je stochastischer diese Beziehungen erscheinen (zum Beispiel aufgrund fehlerbehafteter Daten), desto kleiner muß die Summe der Fehlerquadrate durch das Einfügen einer zusätzlichen Kante werden, um so die zunehmende Komplexität der Struktur zu rechtfertigen.

Implementierung

Auch hier kann grundsätzlich auf die entsprechende Stelle in Abschnitt 3.3.1 verwiesen werden. Es ist zu berücksichtigen, daß bei der Implementierung des Algorithmus für die Anwendung auf ein kontinuierliches DBN die entsprechenden Prozeduren zur Berechnung und Aktualisierung des Scores anzupassen sind:

```
1   PROCEDURE COMPUTE_SCORE(G)
2       score := 0
3       FOR EACH node X_i[t+1] in G
4           score− =COMPUTE_ERROR(i, G)
5       score := score − log M · dimG · σ^2
6       RETURN score

7   PROCEDURE UPDATE_SCORE(G_nb, G_akt, score_akt, i)
8       score_{G_nb} = score_{G_akt}
9       score_{G_nb} + = log M · dimG_akt · σ^2
10      score_{G_nb} − =COMPUTE_ERROR(i, G_akt)
11      score_{G_nb} + =COMPUTE_ERROR(i, G_nb)
12      score_{G_nb} − = log M · dimG_nb · σ^2
13      RETURN score_{G_nb}

14  PROCEDURE COMPUTE_ERROR(i, G)
15      U_G:=U
16      FOR EACH node X_j[t] in G
17          IF edge e=(X_j[t],X_i[t+1])∉ G
18              delete column u_j in U_G
19      compute ŵ^T := (U_G^T U_G)^{-1} U_G^T y_i
20      compute e = ‖y_i − U_G ŵ^T‖
21      RETURN e
```

Der erste Term des BIC-Scores – die Summe der Fehlerquadrate – wird mit Hilfe der Prozedur *COMPUTE_ERROR* (Zeilen 14-21) berechnet. Da nur diejenigen Gewichte betrachtet werden müssen, deren zugehörigen regulatorischen Einflüsse von dem Graphen G beschrieben werden, entfernt diese Prozedur zunächst alle überflüssigen Spaltenvektoren aus der Inputmatrix U (Zeilen 15-18). Anschließend kann dann der Gewichtsvektor w_i mit Hilfe der Methode der kleinsten Quadrate geschätzt (Zeile 19) und der Anteil des Knoten i an der Summe der Fehlerquadrate berechnet (Zeile 20) werden.

Limitationen

Wegen der Anlehnung dieses Modellansatzes an das Additive Regulationsmodell ergeben sich für diesen Ansatz die gleichen modellbedingten Limitationen. Wesentliche

Faktoren sind dabei die Vereinfachung auf ein synchrones, diskretes Zeitsystem, die vereinfachte Annahme einer additiven, unabhängigen Wirkung regulatorischer Einflüsse und die Probleme bei der empirischen Bestimmung der maximalen Expressionsraten.
Ähnlich wie bei den Reverse Engineering Algorithmen für die Additiven Regulationsmodelle beeinflußt die Qualität der Daten die Ergebnisse des Algorithmus; allerdings gelingt es diesem Algorithmus aufgrund der Arbeit mit stochastischen Relationen besser, mit fehlerhaften Daten umzugehen.
Wie schon in Abschnitt 3.3.1 beschrieben, ergibt sich eine algorithmusbedingte Limitation aus der Tatsache, daß der Algorithmus trotz Random-restart Strategie bei der Suche nach einem guten Schätzer für die Struktur G nicht unbedingt die Struktur mit dem global maximalen Score findet und dann nur ein lokales Optimum liefert.

Kapitel 4

Integration von Vorwissen

Sowohl die Komplexität der Genregulationsprozesse als auch die Begrenzung derzeit verfügbarer Expressionsdaten erschweren die Rekonstruktion des den gegebenen Daten zugrundeliegenden Genregulationsnetzwerks. Oftmals ist es allerdings bereits vor dem Reverse Engineering Prozeß möglich, Aussagen über die Wahrscheinlichkeit der Existenz bestimmter regulatorischer Einflüsse zu treffen. Existiert ein solches Vorwissen über die Struktur des zu rekonstruierenden Netzwerks, kann eine Kombination dieses Wissens mit den Informationen aus den gegebenen Expressionsdaten den Reverse Engineering Prozeß wesentlich unterstützen. Die Integration von Vorwissen stellt deshalb eine wichtige Strategie bei der Rekonstruktion von genetischen Netzwerken dar.
Zunächst einmal ist zu klären, wie man Vorwissen überhaupt erlangen kann. Hierfür soll zwischen zwei Möglichkeiten unterschieden werden: Zum einen kann man auf Ergebnisse aus gezielten Experimenten, auf Angaben aus der Literatur und auch auf biologisches Expertenwissen zurückgreifen, um einzelne regulatorische Einflüsse zu charakterisieren. So können zum Beispiel vorangegangene, gezielte Experimente über den regulatorischen Zusammenhang zweier Gene g_i und g_j nahelegen, daß die Expression von g_i durch die Expression von g_j reguliert wird, oder gerade diesen Zusammenhang ausschließen. Auch ein in der Literatur geschilderter regulatorischer Zusammenhang zwischen zwei Genen, der durch gezielte Experimente anderer Arbeitsgruppen aufgedeckt wurde, kann einen von der Existenz eines regulatorischen Einflusses von Gen g_j auf Gen g_i überzeugen. Schließlich könnte man auch aufgrund von biologischem Expertenwissen einen regulatorischen Einfluß zwischen zwei Genen für sehr wahrscheinlich halten oder auch sehr wahrscheinlich ausschließen. Mit diesem Wissen ist es also möglich, über die Existenz bestimmter regulatorischer Einflüsse genaue Aussagen zu treffen und den Reverse Engineering Prozeß diesbezüglich zu unterstützen. Für alle übrigen, nicht betrachteten, aber prinzipiell möglichen regulatorischen Einflüsse können jedoch keine Aussagen gemacht werden. Vor allem in größeren Netzwerken ist die individuelle Betrachtung aller möglichen regulatorischen Einflüsse sehr aufwendig.

Eine zweite Möglichkeit zur Konstruktion von Vorwissen bieten die Ergebnisse aus genomweiten Manipulations- und Expressionsexperimenten. Im Anschluß an die gezielte Manipulation eines Gens g_j in den Zellen einer betrachteten Zellpopulation könnte man nach einem bestimmten Zeitraum Δt ihr Genexpressionsmuster mit dem einer Kontrollpopulation vergleichen, die keinen Manipulationen unterlag. Dieser Vergleich zeigt, welche Gene aufgrund der Manipulation von Gen g_j ihr Expressionsverhalten geändert haben. Wählt man Δt groß genug, ist ein regulatorischer Einfluß von g_j auf g_i relativ sicher nachweisbar, falls er existiert. Sehr wahrscheinlich wird man so aber auch für einige Gene, die nur indirekt von g_j abhängen, ein verändertes Expressionsverhalten aufdecken. Ein in einem solchen Experiment beobachteter Zusammenhang zwischen Gen g_j und g_i läßt deshalb keine sichere Aussage darüber zu, ob ein regulatorischer Einfluß von g_j und g_i tatsächlich existiert. Dafür ist ein regulatorischer Einfluß von Gen g_j auf Gen g_i relativ sicher auszuschließen, wenn keine Reaktion des Gens g_i auf die Manipulation von Gen g_j zu beobachten war. Mit den Ergebnissen aus derartigen Experimenten ist es damit zwar nicht möglich, die Existenz bestimmter regulatorischer Einflüsse im Reverse Engineering Prozeß zu unterstützen, aber vor allem in größeren Netzen kann man mit dem gewonnenen Wissen sehr viele der potentiellen regulatorischen Einflüsse ausschließen und so das Reverse Engineering vereinfachen.
In der Literatur finden sich hauptsächlich Ansätze zur Integration von Vorwissen bei der Arbeit mit (Dynamischen) Bayesschen Netzwerken. Aber auch Reverse Engineering Methoden, die auf Booleschen Netzwerken oder Additiven Regulationsmodellen basieren, können durch Vorwissen unterstützt werden. Dieses Kapitel soll im folgenden erläutern, wie die im vorangegangenen Kapitel 3 zu den Netzwerkmodellen Boolesche Netzwerke, (diskrete und kontinuierliche) Dynamische Bayessche Netzwerke und Additive Regulationsmodelle vorgestellten Reverse Engineering Algorithmen vorhandenes Vorwissen in den Rekonstruktionsprozeß integrieren können.

4.1 Boolesche Netzwerke

Einen wichtigen Ansatz für das Reverse Engineering in Booleschen Netzwerken stellt der Algorithmus *Reveal* dar. Dieser Algorithmus ist in seiner ursprünglichen Form nicht für die Arbeit mit fehlerbehafteten Expressionsdaten geeignet. Wie in Abschnitt 3.2.1 beschrieben, orientiert sich die vorliegenden Arbeit deshalb an einer erweiterten Variante des Algorithmus, die von [42, 46] speziell an den Umgang mit fehlerbehafteten und unvollständigen Trainingsdaten angepaßt wurde. Auf Grundlage der beiden Eigenschaften

$$MI(X, x_i) \cdot M \cdot \ln 4 \sim \chi^2_{df,1-\alpha_1} \tag{4.1}$$

$$(MI(X, x_i) - MI(X \backslash z, x_i)) \cdot M \cdot \ln 4 \sim \chi^2_{df;1-\alpha_2} \tag{4.2}$$

konnten hier mit Hilfe der wechselseitigen Information und χ^2-Unabhängigkeitstests Inputelemente als Elternelemente des Outputelements x_i identifiziert und so die auf das Gen g_i einwirkenden regulatorischen Einflüsse festgelegt werden.
Bei der Arbeit mit der wechselseitigen Information ist es nicht möglich, Vorwissen über die Struktur des Genregulationsnetzwerks zu integrieren. Deshalb schlägt [42] vor, auf die in [41] bewiesene Eigenschaft

$$-2 \ln LR(X, x_i) = MI(X, x_i) \cdot M \cdot \ln 4 \tag{4.3}$$

zurückzugreifen. Äquivalent zur wechselseitigen Information $MI(X, x_i)$ kann also auch mit der Likelihood Ratio $LR(X, x_i)$ gearbeitet werden. Beide liefern ein Maß dafür, wie stark das Outputelement x_i von der Menge X an Inputelementen abhängt. Die wechselseitige Information bestimmt hierzu den Anteil der Information, die dem Outputelement x_i und der Menge X gemeinsam ist, und definiert darüber die Stärke der Abhängigkeit. Die Likelihood Ratio $LR(X, x_i)$ dagegen betrachtet die beiden Punkthypothesen

$$H_{0,(X,x_i)}: \quad x_i \text{ ist unabhängig von } X$$

$$H_{1,(X,x_i)}: \quad x_i \text{ ist abhängig von } X$$

und liefert unter Berücksichtigung der gegebenen Daten D die relative Evidenz (Plausibilität) der Hypothesen zueinander. Dazu stellt sie vergleichend die Wahrscheinlichkeit der gegebenen Daten unter der Hypothese $H_{0,(X,x_i)}$ der Wahrscheinlichkeit der gegeben Daten unter der Hypothese $H_{1,(X,x_i)}$ gegenüber:

$$LR(X, x_i) = \frac{P(D|H_{0,(X,x_i)})}{P(D|H_{1,(X,x_i)})} \tag{4.4}$$

Eine Likelihood Ratio größer dem Wert 1 besagt, daß die gegebenen Daten unter der Hypothese $H_{0,(X,x_i)}$ wahrscheinlicher sind als unter der Hypothese $H_{1,(X,x_i)}$ und liefert so Evidenz für $H_{0,(X,x_i)}$ gegenüber $H_{1,(X,x_i)}$. Diese Evidenz ist umso höher, je größer die Likelihood Ratio ist. Analog bedeutet eine Likelihood Ratio kleiner dem Wert 1 Evidenz für $H_{1,(X,x_i)}$ gegenüber $H_{0,(X,x_i)}$, die um so höher ausfällt, je kleiner die Likelihood Ratio ist. Die Likelihood Ratio kann damit interpretiert werden als das, was die Daten relativ zu $H_{0(X,x_i)}$ und $H_{1,(X,x_i)}$ sagen.
Die Arbeit mit der Likelihood Ratio erfolgt analog zur Arbeit mit der wechselseitigen Information: Um die Elternelemente für das Outputelement x_i festzulegen, wird zunächst für jede beliebige Menge X aus maximal k_{max} Inputelementen mit dem ersten χ^2-Unabhängigkeitstest überprüft, ob eine Unabhängigkeit des Outputelements x_i von der Menge X abgelehnt werden kann. Die Eigenschaft 4.1 gilt analog auch für die Likelihood Ratio [41]:

$$-2 \ln LR \sim \chi^2_{df,1-\alpha_1} \tag{4.5}$$

Liefert die Likelihood Ratio $LR(X, x_i)$ genügend große Evidenz für $H_{1,(X,x_i)}$ gegenüber $H_{0,(X,x_i)}$, darf man die Unabhängigkeit von x_i und X ablehnen:

$$-2 \ln LR(X, x_i) > \chi^2_{df;1-\alpha_1} \tag{4.6}$$

Besteht die Menge X nur aus einem einzigen Inputelement, kann dieses anschließend direkt in die Elternmenge von x_i aufgenommen werden. Andernfalls ist für jedes Inputelement z in X zusätzlich ein zweiter χ^2-Unabhängigkeitstest heranzuziehen. Dieser untersucht, ob das jeweilige Inputelement z wirklich benötigt wird, um die Abhängigkeit des Outputelements x_i von der Menge X festzulegen, oder ob x_i eigentlich nur von der Teilmenge $X \backslash z$ abhängt. Dafür läßt sich auch die Eigenschaft 4.2 auf die Likelihood Ratio übertragen:

$$-2(\ln LR(X, x_i) - \ln LR(X \backslash z, x_i)) \sim \chi^2_{df;1-\alpha_2} \tag{4.7}$$

Die Nullhypothese, daß das Inputelement z nicht benötigt wird, um die Abhängigkeit des Outputelements x_i von der Menge X zu definieren, kann deshalb abgelehnt werden, falls:

$$-2(\ln LR(X, x_i) - \ln LR(X \backslash z, x_i)) > \chi^2_{df,1-\alpha_2} \tag{4.8}$$

Das Inputelement z gehört dann in die Elternmenge von x_i.
Wie läßt sich nun bei der Arbeit mit Likelihood Ratios vorhandenes Vorwissen über die Struktur des Genregulationsnetzwerks integrieren? Zunächst einmal ist es für jedes Paar zweier Netzwerkkomponenten x_i und x_j erforderlich, eine Wahrscheinlichkeit $P(H_{1,(x_j,x_i)})$ für die Hypothese $H_{1,(x_j,x_i)}$ festzulegen, daß x_j einen regulatorischen Einfluß auf x_i ausübt. Diese Wahrscheinlichkeiten spiegeln das vorhandene Vorwissen wider. Ist man sich relativ sicher, daß das Gen g_j das Gen g_j reguliert, sollte eine hohe Wahrscheinlichkeit $P(H_{1,(x_j,x_i)})$ gewählt werden. Umgekehrt entscheidet man sich für eine entsprechend kleine Wahrscheinlichkeit $P(H_{1,(x_j,x_i)})$, wenn ein regulatorischer Einfluß von Gen g_j auf Gen g_i relativ sicher ausgeschlossen werden kann. Existiert für den regulatorischen Einfluß von Gen g_j auf Gen g_i kein Vorwissen, ist der Wahrscheinlichkeit $P(H_{1,(x_j,x_i)})$ der Wert 0.5 zuzuordnen. Solche Vermutungen über die Existenz der einzelnen regulatorischen Einflüsse werden in [42] als zusätzliche Beobachtungen verstanden. Sie definieren Prior-Wahrscheinlichkeiten für die Hypothesen $H_{0,(X,x_i)}$ und $H_{1,(X,x_i)}$, daß eine Menge X von Inputelementen und ein Outputelement x_i unabhängig bzw. abhängig voneinander sind: Besteht die Menge X lediglich aus einem Inputelement y, dann ergeben sich die Prior-Wahrscheinlichkeiten $P(H_{1,(X,x_i)})$ und $P(H_{0,(X,x_i)})$ aus den Wahrscheinlichkeiten $P(H_{1,(y,x_i)})$ und $(1 - P(H_{1,(y,x_i)}))$. Ist die Anzahl der Elemente in X größer, sind die Prior-Wahrscheinlichkeiten $P(H_{1,(X,x_i)})$ und $P(H_{0,(X,x_i)})$ durch das Inputelement y mit der größten Einzelwahrscheinlichkeit $P(H_{1,(y,x_i)})$ definiert. Im Reverse Engineering Prozeß werden die Prior-Wahrscheinlichkeiten für $H_{0,(X,x_i)}$ und $H_{1,(X,x_i)}$ an-

schließend durch die gegebenen Daten aktualisiert, und es lassen sich mit Hilfe des Theorems von Bayes die entsprechenden Posterior-Wahrscheinlichkeiten formulieren:

$$P(H_{*,(X,x_i)}|D) = \frac{P(D|H_{*,(X,x_i)}) \cdot P(H_{*,(X,x_i)})}{P(D)} \tag{4.9}$$

Die Likelihood Ratio $LR(X, x_i)$ entspricht jetzt dem Quotienten dieser Posterior-Wahrscheinlichkeiten:

$$LR(X, x_i) = \frac{P(H_{0,(X,x_i)}|D)}{P(H_{1,(X,x_i)}|D)} = \frac{P(D|H_{0,(X,x_i)})}{P(D|H_{1,(X,x_i)})} \cdot \frac{P(H_{0,(X,x_i)})}{P(H_{1,(X,x_i)})} \tag{4.10}$$

Abschließend soll kurz auf die Berechnung der Likelihood Ratio eingegangen werden: Sei M der Umfang der gegebenen Daten D und M_{k_i} die Anzahl der Beobachtungen in D, in denen das Outputelement x_i den Wert k_i aufweist. M_{j_i} beschreibt analog, wie häufig die Wertekombination j_i für die Menge X beobachtet wurde. Entsprechend gibt M_{k_i,j_i} die Anzahl der Beobachtungen an, bei denen eine Kombination der Werte k_i für das Outputelement x_i und j_i für die Menge X zu verzeichnen ist. Die Likelihood Ratio $LR(X, x_i)$ ergibt sich dann aus [41]:

$$LR(X, x_i) = \frac{\prod_{j_i} P(X = j_i)^{M_{j_i}} \cdot \prod_{k_i} P(x_i = k_i)^{M_{k_i}}}{\prod_{j_i,k_i} P(X = j_i, x_i = k_i)^{M_{j_i,k_i}}} \cdot \frac{P(H_{0,(X,x_i)})}{P(H_{1,(X,x_i)})} \tag{4.11}$$

Die entsprechenden Wahrscheinlichkeiten können dabei empirisch aus den Daten mit Hilfe der Maximum Likelihood Schätzung bestimmt werden:

$$\begin{aligned} P(x_i = k_i) &= \frac{M_{k_i}}{M} \\ P(X = j_i) &= \frac{M_{j_i}}{M} \\ P(X = j_i, x_i = k_i) &= \frac{M_{k_i,j_i}}{M} \end{aligned} \tag{4.12}$$

4.2 Additive Regulationsmodelle

Die Aufgabe eines Reverse Engineering Algorithmus für das Additive Regulationsmodell ist es, einen möglichst guten Schätzer für die Gewichtsmatrix W zu finden. Alle in Kapitel 3 für dieses genetische Netzwerkmodell vorgestellten Algorithmen gehen dabei zunächst von einem vollständig verknüpften Netzwerk aus. Es sind dann geeignete Methoden erforderlich, die die Schätzer $\hat{w}_{ij}$ der einzelnen Gewichte w_{ij} in „null“ und „nicht-null“ klassifizieren (vergleiche Abschnitt 3.4). Die Struktur des

Netzwerks wird damit nur implizit über die Schätzung der Gewichtsmatrix erlernt und der Reverse Engineering Prozeß zusätzlich erschwert.
Vorwissen könnte hier die Rekonstruktion des Genregulationsnetzwerks dadurch unterstützen, indem alle Gewichte w_{ij}, von denen man annimmt, daß der zugehörige regulatorische Einfluß von Gen g_j auf Gen g_i im zugrundeliegendem Genregulationsnetzwerk nicht existiert, auf den Wert 0 festgelegt werden. Ein Reverse Engineering Algorithmus muß dann weniger Gewichte schätzen und kann so bei gleichem Umfang der gegebenen Daten die einzelnen Schätzer mit einer größeren Genauigkeit bestimmen. Dies wiederum wirkt sich positiv auf eine korrekte Klassifizierung der Schätzer aus.
Gleichzeitig könnte man die Klassifizierung der Schätzer zusätzlich dadurch unterstützen, daß die Gewichte w_{ij} aller sicher existierenden regulatorischen Einflüsse zwingend in die Klasse „nicht-null“ eingeordnet werden.
Nachteilig ist hier die Tatsache, daß im Gegensatz zu dem vorangegangenen Ansatz für Boolesche Netzwerke dieser Ansatz keine Möglichkeit bietet, qualitative Angaben über die Sicherheit des Vorwissen zu integrieren. Die Integration der Annahme, daß ein regulatorischer Einfluß zwischen zwei Netzwerkkomponenten wahrscheinlich nicht existiert, bedeutetet immer, daß das rekonstruierte Netzwerk diesen Einfluß auch nicht repräsentieren wird, unabhängig von der Information, welche die Daten liefern. Deshalb sollte hier nur sehr sicheres Wissen über die Existenz von regulatorischen Einflüssen integriert werden.

4.3 Dynamische Bayessche Netzwerke

Verwendet man zur Modellierung des Genregulationsnetzwerks ein Dynamisches Bayessches Netzwerk $B = \langle G, \Theta_G \rangle$, muß ein Reverse Engineering Algorithmus zunächst einen guten Schätzer für die Struktur G erlernen. Eine weitere Aufgabe besteht darin, die Parametermenge Θ, die für jede Zufallsvariable des Netzwerks eine entsprechende bedingte Wahrscheinlichkeitsverteilung definiert, aus den gegebenen Trainingsdaten zu schätzen.
Diskrete und kontinuierliche DBN können in diesem Abschnitt gemeinsam betrachtet werden, denn das Erlernen der Struktur, das man durch die Integration von Vorwissen über die Struktur des Genregulationsnetzwerks unterstützen möchte, erfolgte in dieser Arbeit bei diskreten und kontinuierlichen DBN nach dem gleichen Prinzip: Ausgehend von einer bestimmten Startstruktur wurde iterativ durch das Einfügen oder Löschen einer Kante eine neue Struktur mit einem besseren BIC-Score erzeugt. Die Herleitung des BIC-Scores erfolgte über die Posterior-Wahrscheinlichkeit $P(G|D)$ einer Struktur bezüglich der gegebenen Daten D (vergleiche Unterabschnitt *Scoring-Funktionen* in Abschnitt 3.3):

$$Score(G|D) = P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{4.13}$$

Die Wahrscheinlichkeit $P(D)$ der Daten konnte als strukturunabhängige Konstante vernachlässigt werden. Ebenso entfiel bei den bisherigen Betrachtungen die Prior-Wahrscheinlichkeit $P(G)$, denn ist kein Vorwissen über die Struktur des Genregulationsnetzwerks vorhanden, kann eine nichtinformative Prior-Wahrscheinlichkeitsverteilung angenommen werden und $P(G)$ reduziert sich ebenfalls zu einer strukturunabhängigen Konstante. Möchte man nun vorhandenes Vorwissen über die Struktur des Genregulationsnetzwerks integrieren, muß man die Prior-Wahrscheinlichkeit $P(G)$ an dieser Stelle beibehalten. Sie ist dann im resultierenden BIC-Score als zusätzlicher Summand zu berücksichtigen:

$$Score_{BIC}(G, D) = \log P(D|G, \hat{\Theta}_G) - \frac{\log M}{2} dimG + \log(P(G)) \tag{4.14}$$

Der erste Term – die Likelihood Funktion – muß wie gehabt jeweils geeignet an das diskrete bzw. kontinuierliche DBN angepaßt werden. Außerdem ist es jetzt erforderlich, die Prior-Wahrscheinlichkeitsverteilung der Strukturen mit Hilfe des Vorwissens festzulegen:
Der einfachste Ansatz hierbei ordnet allen Strukturen G, die sicher nicht existierende regulatorische Einflüsse modellieren, eine Wahrscheinlichkeit $P(G)$ vom Wert 0 zu [25]. Die Wahrscheinlichkeitsverteilung über die verbleibenden Strukturen wird als uniform betrachtet. Die praktische Umsetzung erfolgt durch eine Einschränkung des Suchraums S. Alle Strukturen, die sicher nicht existierende regulatorische Einflüsse beschreiben, werden aus dem Suchraum ausgeschlossen. Da die Prior-Wahrscheinlichkeit für die verbleibenden Strukturen uniform ist, muß man sie dann bei der Berechnung des BIC-Scores analog zur bisherigen Verfahrensweise nicht berücksichtigen.
Wie schon für den vorangegangenen Ansatz zur Integration von Vorwissen in Additiven Regulationsmodellen geschildert, sollte man hier nur sehr sicheres Vorwissen über die Existenz von regulatorischen Einflüssen integrieren, denn die Integration der Annahme, daß ein bestimmter regulatorischer Einfluß zwischen zwei Netzwerkkomponenten wahrscheinlich nicht existiert, bedeutet, daß dieser auch nicht in dem rekonstruierten Netzwerk enthalten sein wird.
Eine andere interessante Möglichkeit zur Definition der Prior-Wahrscheinlichkeiten $P(G)$, die es auch erlaubt, qualitative Angaben über die Sicherheit des vorhandenen Vorwissens zu berücksichtigen, läßt sich in [25] finden: Zunächst konstruiert man die bezüglich des Vorwissens wahrscheinlichste Struktur G^*. Für die Berechnung der Prior-Wahrscheinlichkeit einer beliebigen Struktur G ist dann für jede der N Zufallsvariablen $X_i(t+1)$ die Anzahl δ_i der Einträge zu bestimmen, in denen sich die Elternmengen Pa von $X_i(t+1)$ in G und G^* voneinander unterscheiden:

$$\delta_i = (Pa_G \cup Pa_{G^*}) \backslash (Pa_G \cap Pa_{G^*}) \tag{4.15}$$

Die Prior-Wahrscheinlichkeit von G ergibt sich anschließend aus

$$P(G) = \kappa^{\sum_{i=1}^{N} \delta_i} = \prod_{i=1}^{N} \kappa^{\delta_i} \tag{4.16}$$

Die Konstante κ ist dabei auf einen Wert aus dem Intervall $(0, 1]$ festgesetzt. Sie legt den Einfluß fest, den die Prior-Wahrscheinlichkeit $P(G)$ auf den Gesamtwert des Scores ausübt, und bestimmt damit, wie stark das Vorwissen im Vergleich zu den gegebenen Daten im Reverse Engineering Prozeß berücksichtigt wird. Deshalb sollte ihr Wert die Sicherheit über die Struktur G^* widerspiegeln. Ein Wert nahe 1 sagt aus, daß man sich bezüglich der Korrektheit von G^* nicht sehr sicher ist. Der Anteil der Prior-Wahrscheinlichkeit $P(G)$ einer Struktur G an ihrem Score fällt dann recht klein aus, und das Vorwissen hat im Verhältnis zu den gegebenen Daten im Reverse Engineering Prozeß kaum Einfluß. Umgekehrt bewirkt ein Wert nahe 0 einen im Verhältnis zu den gegebenen Daten recht großen Einfluß des Vorwissens auf den Reverse Engineering Prozeß und besagt, daß man sehr überzeugt von der Struktur G^* ist.
Kann man für einige Gene über die auf sie wirkenden regulatorischen Einflüsse ein detaillierteres und sichereres Vorwissen nachweisen als für andere, ist es weiterhin möglich, für jede Zufallsvariable $X_i(t + 1)$ eine eigene Konstante κ_i zu definieren [25]. Die Prior-Wahrscheinlichkeit $P(G)$ ergibt sich dann analog zu 4.16 aus:

$$P(G) = \prod_{i=1}^{N} \kappa_i^{\delta_i} \tag{4.17}$$

In dieser Form bietet der Ansatz allerdings ebenfalls keine Möglichkeit, qualitative Angaben über die Sicherheit des Vorwissens einzelner regulatorischer Einflüsse zu berücksichtigen. Dies kann sehr nachteilig sein. Ist man sich zum Beispiel sehr sicher, daß eine Netzwerkkomponente $X_i(t+1)$ von der Netzwerkkomponente $X_j(t)$ reguliert wird, kann aber andere Einflüsse auf $X_i(t+1)$ nicht ausschließen, ergibt sich folgendes Problem: Wählt man einen kleinen Wert für κ_i, ist es im Reverse Engineering Prozeß kaum möglich, weitere regulatorische Einflüsse auf $X_i(t + 1)$ aufzudecken. Ein Wert für κ_i nahe 1 dagegen erlaubt es wiederum nicht, das Wissen über die Existenz des regulatorischen Einflusses von $X_j(t)$ auf $X_i(t + 1)$ im Reverse Engineering Prozeß adäquat zu berücksichtigen.
Für diese Arbeit soll der Ansatz deshalb entsprechend erweitert werden und so auch ermöglichen, qualitative Angaben über die Sicherheit des Vorwissens einzelner regulatorischer Einflüsse in den Reverse Engineering Prozeß zu integrieren. Dabei muß für jeden potentiellen regulatorischen Einfluß eine eigene Konstante κ_{ij} spezifiziert werden, die die Sicherheit des Vorwissens über seine Existenz beschreibt. Um die Prior-Wahrscheinlichkeit für eine beliebige Struktur G festzulegen, bestimmt man dann für jeden einzelnen potentiellen regulatorischen Einfluß den Parameter δ_{ij}, der

angibt, ob G und G^* bezüglich dieses Einflusses übereinstimmen, oder ob sie sich widersprechen. Im ersten Fall nimmt δ_{ij} den Wert 1 an, im zweiten Fall den Wert 0. In Anlehnung an 4.17 ergibt sich die Prior-Wahrscheinlichkeit $P(G)$ dann aus:

$$P(G) = \prod_{i=1}^{N} \prod_{j=1}^{N} \kappa_{ij}^{\delta_{ij}} \tag{4.18}$$

Kapitel 5

Experimente an Simulationsdaten

Alle in Kapitel 3 vorgestellten Reverse Engineering Algorithmen wurden implementiert und zunächst an Simulationsdaten getestet, die von zufällig erzeugten Netzwerken generiert wurden. Die aus diesen Experimenten resultierenden Ergebnisse sollen in diesem Kapitel ausführlich analysiert und diskutiert werden. Die Motivation für die Arbeit mit Simulationsdaten liefert die Tatsache, daß die ihnen zugrundeliegenden Netzwerke bekannt sind und die von den einzelnen Algorithmen rekonstruierten Netzwerke genau evaluiert werden können.
Zur Erinnerung faßt Tabelle 5.1 alle Reverse Engineering Methoden zusammen. Neben dem entsprechenden Netzwerkmodell gibt sie den Reverse Engineering Algorithmus an, der auf Basis der gegebenen Expressionsdaten die Modellparameter festlegt und beschreibt kurz seine Grundidee. Außerdem ist für jeden Algorithmus die Farbe aufgelistet, die ihn in den Diagrammen der folgenden Analysen repräsentiert. Der Algorithmus *Reveal* wurde mit zwei verschiedenen Belegungen der Signifikanzniveaus α_1 und α_2 getestet, um den Einfluß dieser Parameter zu verdeutlichen. Bei der Betrachtung des linearen Additiven Regulationsmodells beschränken sich die folgenden Analysen auf den *REM* Algorithmus.
Dieses Kapitel schildert zunächst einführend die Erzeugung der zufälligen Modellnetzwerke und die Generierung der Simulationsdaten. Außerdem werden die Evaluierungsmaße eingeführt, die zur Berechnung der Güte eines rekonstruierten Netzwerks und damit zur Bewertung des entsprechenden Algorithmus dienten. Es folgt eine detaillierte Diskussion über die Abhängigkeit der Algorithmen von verschiedenen Eigenschaften der gegebenen Daten und der diesen Daten zugrundeliegenden Netzwerke. Anschließend überprüft dieses Kapitel, ob es durch die Kombination der Ergebnisse zweier Algorithmen möglich ist, diese zu verbessern. Den Abschluß bildet ein Experiment zur Integration von Vorwissen.
Die Experimente in diesem Kapitel sollen vor allem aufzeigen, welche Ergebnisse die einzelnen Algorithmen in Abhängigkeit von verschiedenen Parametern des zu rekonstruierenden Netzwerks und der gegebenen Daten liefern.

Reverse Engineering Algorithmen	Netzwerkmodelle	Grundidee	Repräsentation
Adjazenzlisten-Konstruktion	Gerichteter Graph	Generiere Erreichbarkeitsliste aus den Daten und verkleinere diese schrittweise zur Adjazenzliste des minimalen Graphen.	
Reveal	Boolesches Netzwerk	Benutze die wechselseitige Information und statistische Tests, um eine Menge aus k Inputelementen als Elternmenge eines Outputelements x_i festzulegen.	α_1=0.01 α_2=0.001 α_1=0.001 α_2=0.0001
Strukturlernen in diskreten DBN	Diskretes DBN	Durchsuche den Raum aller möglichen Strukturen nach einem guten Schätzer der wahren Struktur. Ausgehend von einer Startstruktur erzeuge iterativ durch das Einfügen oder Löschen einer Kante eine neue Struktur mit einem höheren BIC-Score.	
Reverse Engineering in Matrizen - REM	Lineares Additives Regulationsmodell	Bestimme einen Schätzer der Gewichtsmatrix W mit Hilfe der Methode der kleinsten Quadrate.	
Reverse Engineering in Matrizen - REM	Nichtlineares Additives Regulationsmodell	Bestimme einen Schätzer der Gewichtsmatrix W mit Hilfe der Methode der kleinsten Quadrate.	
Evolutionärer Algorithmus	Nichtlineares Additives Regulationsmodell	Bestimme einen Schätzer der Gewichtsmatrix W mit Hilfe eines stochastischen, an der biologischen Evolution orientierten Optimierungsverfahrens.	
Backpropagation Through Time -BPTT	Nichtlineares Additives Regulationsmodell	Bestimme einen Schätzer der Gewichtsmatrix W mit Hilfe eines deterministischen, gradientenbasierten Optimierungsverfahrens.	
Strukturlernen in kontinuierlichen DBN	Kontinuierliches DBN	Durchsuche den Raum aller möglichen Strukturen nach einem guten Schätzer der wahren Struktur. Ausgehend von einer Startstruktur erzeuge iterativ durch das Einfügen oder Löschen einer Kante eine neue Struktur mit einem höheren, an das kontinuierliche Modell angepaßten BIC-Score.	

Tabelle 5.1: **Reverse Engineering Methoden im Überblick.**

5.1 Experimentelles Design

5.1.1 Erzeugung der Modellnetzwerke

Zur Generierung der Simulationsdaten sind geeignete Modellnetzwerke erforderlich. Als Grundlage für die Erzeugung zufälliger Modellnetzwerke muß ein genetisches Netzwerkmodell ausgewählt werden. In dieser Arbeit fiel die Wahl in Anlehnung an [13, 64] auf das nichtlineare Additive Regulationsmodell (siehe Abschnitt 2.2.4). Die auf Basis dieses Netzwerkmodells erzeugten Netzwerke abstrahieren stark von der biologischen Realität. Bei der Arbeit mit den von ihnen generierten Simulationsdaten kommen deshalb viele der modell- und algorithmenbedingten Limitationen der Reverse Engineering Algorithmen nicht zum Tragen, und die in den Experimenten erzielten Ergebnisse der einzelnen Algorithmen lassen sich nicht direkt auf die Arbeit mit realen Expressionsdaten übertragen. Es ist jedoch zu hoffen, daß die Experimente mit solchen Simulationsdaten Einblicke in das grundlegende Verhalten der Algorithmen in Abhängigkeit von verschiedenen Eigenschaften der Netzwerke und der verfügbaren Daten ermöglichen.
Der in dieser Arbeit verwendete Algorithmus zur Erzeugung eines zufälligen Netzwerks der Größe N und der Konnektivität k besteht im wesentlichen aus zwei Teilen:

Schritt 1 – Generiere die Struktur:
Erzeuge N Netzwerkkomponenten x_i, welche die Expressionsraten der Gene eines Genregulationsnetzwerks modellieren. Da alle Algorithmen ausführlich untersucht werden sollten, wurde in dieser Arbeit mit relativ kleinen Netzwerken gearbeitet, um den Rechenaufwand des Reverse Engineerings im Rahmen zu halten.
Erzeuge anschließend die regulatorischen Einflüsse. Für jede Netzwerkkomponente x_i wähle dazu zufällig k verschiedene Elemente aus der Menge aller Netzwerkkomponenten aus. Diese bilden dann die Elternelemente von x_i.

Schritt 2 – Generiere die Dynamik:
Wähle für jede Netzwerkkomponente x_i zufällig einen ganzzahligen Wert aus dem Intervall $[min, max]$ aus und ordne ihn ihrer maximalen Expressionsrate max_i zu. Das Intervall betrug in dieser Arbeit immer $[1, 10]$.
Es folgt die Belegung der Gewichtsvektoren w_i, um die regulatorischen Einflüsse genauer zu spezifizieren. Für alle Netzwerkkomponenten x_k, die keinen regulatorischen Einfluß auf die Netzwerkkomponente x_i ausüben, setze die zugehörigen Gewichte w_{ik} auf den Wert 0. Für alle anderen Gewichte w_{ij} generiere jeweils eine normalverteilte, von dem Wert 0 verschiedene Zufallszahl: $w_{ij} \sim N(0, 10)$, $w_{ij} \neq 0$. Jede Netzwerkkomponente x_i sollte alle Werte aus dem Intervall $(0, max_i)$ annehmen können. Zur Überprüfung dieser Eigenschaft berechne jeweils den maximalen regulatorischen Input $r_{i,max}$. Er ergibt sich, falls alle Netzwerkkomponenten, die einen positiven regulatorischen Einfluß auf x_i ausüben, einen Wert nahe ihrer maximalen Expressionsrate

angenommen haben, und alle Netzwerkkomponenten, die negativ auf x_i einwirken, einen Wert nahe 0 aufweisen. Berechne analog den minimalen regulatorischen Input $r_{i,min}$:

$$\begin{aligned} r_{i,max} &= \sum_{j,w_{ij}>0} w_{ij} \cdot max_j \\ r_{i,min} &= \sum_{j,w_{ij}<0} w_{ij} \cdot max_j \end{aligned} \tag{5.1}$$

Es sollte gelten:

$$\begin{aligned} \frac{max_i}{1+e^{-(r_{i,max}+\beta_i)}} &\approx max_i \\ \frac{max_i}{1+e^{-(r_{i,min}+\beta_i)}} &\approx 0 \end{aligned} \tag{5.2}$$

Dafür müssen die Exponenten $(r_{i,max}+\beta_i)$ und $(r_{i,min}+\beta_i)$ betragsmäßig hinreichend groß sein. In dieser Arbeit wurde der Wert 10 als hinreichend groß betrachtet:

$$\begin{aligned} r_{i,max} + \beta_i &> 10 \\ r_{i,min} + \beta_i &< -10 \end{aligned} \tag{5.3}$$

Versuche, durch eine geeignete Wahl des Biasfaktors β_i diese Gleichungen zu erfüllen:

FALLS $(r_{i,max} > 10 \wedge r_{i,min} < -10)$
$\rightarrow \beta_i = 0$

FALLS $(r_{i,max} > 10 \wedge r_{i,min} > -10)$
FALLS $(r_{i,max} + (-10 - r_{i,min}) > 10)$
$\rightarrow \beta_i = -10 - r_{i,min}$

FALLS $(r_{i,max} < 10 \wedge r_{i,min} < -10)$
FALLS $(r_{i,min} + (10 - r_{i,max}) < -10)$
$\rightarrow \beta_i = 10 - r_{i,max}$

FALLS $(r_{i,max} < 10 \wedge r_{i,min} > -10)$
$\rightarrow$ verwerfe w_i

Ist dies nicht möglich, verwerfe alle von dem Wert 0 verschiedenen Gewichte w_{ij} und versuche erneut, sie geeignet zu belegen.

5.1.2 Generierung der Simulationsdaten

Nach der Erzeugung eines zufälligen Netzwerks konnten im Anschluß daran mit Hilfe der in Kapitel 3 beschriebenen Expressions- und Manipulationsexperimente (Definitionen 3.1 und 3.2) eine Zeitreihe, Zustandsübergangsdaten und stabile Zustandsdaten (Definitionen 3.4, 3.3 und 3.5) generiert werden. Der Startzustand des Netzwerks für die Generierung einer Zeitreihe sowie die jeweiligen Anfangszustände der Zustandsübergangspaare von Zustandsübergangsdaten wurden dabei zufällig gewählt. Die entsprechenden Aktualisierungsregeln (Gleichung 2.11) dienten zur Modellierung von Zustandsübergängen. Manipulationsexperimente wurden durchgeführt, nachdem das Netzwerk, ausgehend von einem zufällig gewählten Startzustand, einen Attraktor erreicht hatte. Nach Definition 2.5 ist ein Netzwerk in einem Attraktor angelangt, falls es sich in einem Zustand $S(t_n)$ befindet, den es bereits einmal zu einem vorangegangenen Zeitpunkt t_m angenommen hatte. Ein diesbezüglicher Test fiel in der für diese Arbeit implementierten Realisierung positiv aus, wenn sich die Werte von $S(t_m)$ und $S(t_n)$ bis auf drei Stellen nach dem Komma glichen.
Es wurden nur solche Netzwerke betrachtet, die ein interessantes, dynamisches Verhalten zeigten und nicht sofort in einen Attraktor gelangten. Dies sollte absichern, daß die jeweils generierte Zeitreihe eine Mindestanzahl an verschiedenen Zustandsübergängen aufweist und damit genügend Information für die Rekonstruktion des Netzwerks bereitstellt. Die benötigte Mindestanzahl an Zustandsübergängen hängt von dem Typ des zu rekonstruierenden Netzwerks sowie von den Parametern Netzwerkgröße und Konnektivität ab. Entsprechende Abschätzungen (siehe u.a. [57]) besagen, daß beispielsweise für die Rekonstruktion eines Booleschen Netzwerks der Größe N und der Konnektivität k

$$k \cdot 2^k \cdot \log(N) \tag{5.4}$$

verschiedene Zustandsübergänge benötigt werden. Beschränken sich die Booleschen Funktionen des Netzwerks auf linear separierbare Regeln, verringert sich dies Anzahl auf

$$k \cdot \log(N/k) \tag{5.5}$$

Linear separierbare Funktionen zeichnen sich dadurch aus, daß für jeden regulatorischen Einfluß eindeutig festgelegt ist, ob er einen positiven oder einen negativen Effekt hat. Bei nichtlinear separierbaren Regeln hingegen kann ein regulatorischer Einfluß in Abhängigkeit von allen anderen, auf dieselbe Netzwerkkomponente einwirkenden regulatorischen Einflüssen sowohl einen positiven als auch einen negativen Effekt zeigen [1].

[1] Ein Beispiel für eine nichtlinear separierbare Boolesche Regel ist die XOR Funktion.

Die von dem Additiven Regulationsmodell zur Modellierung regulatorischer Einflüsse verwendeten Funktionen sind ebenfalls linear separierbar und bilden das Analogon zu linear separierbaren Booleschen Regeln. Eine Herleitung der benötigten Datenmenge für diesen Netzwerktyp in [27] liefert eine zu den linear separierbaren Booleschen Netzwerken identische Abschätzung (siehe 5.5).
Wieviele verschiedene Zustandsübergänge werden nun konkret von den in dieser Arbeit betrachteten Reverse Engineering Algorithmen benötigt? Der *Reveal* Algorithmus versucht, mit Hilfe der aus einem Additiven Regulationsnetzwerk generierten Simulationsdaten ein Boolesches Netzwerk zu konstruieren. Die Simulationsdaten müssen dazu entsprechend diskretisiert werden. Dabei geht sehr viel Information über das dynamische Verhalten des Netzwerks verloren; es entstehen Fehler und Inkonsistenzen. Deshalb kann hier nicht auf die Abschätzungen 5.4 sowie 5.5 zurückgegriffen werden, und es ist unklar, wieviele verschiedene Zustandsübergänge der *Reveal* Algorithmus tatsächlich benötigt. Aus dem gleichen Grund können auch für den Algorithmus *Strukturlernen in diskreten DBN* keine Vorhersagen über den benötigten Datenumfang getroffen werden.
Da für die auf diskreten Netzwerkmodellen basierenden Reverse Engineering Methoden keine Aussage über die Mindestanzahl benötigter Zustandsübergänge möglich ist, orientierte sich diese Arbeit an Reverse Engineering Methoden, die ein Additives Regulationsmodell voraussetzen, um die Mindestanzahl an verschiedenen Zustandsübergängen in einer Zeitreihe festzulegen. Allerdings kann auch hier nicht mit der Abschätzung $k \cdot \log(N/k)$ gearbeitet werden. Die Algorithmen *REM* und *BPTT* sowie der *evolutionäre Algorithmus* versuchen zwar, das den Simulationsdaten zugrundeliegende, linear separierbare Additive Regulationsnetzwerk auch durch ein solches zu modellieren. Sie gehen aber bei der Rekonstruktion des Netzwerkes von einem vollständig verknüpften Netzwerk aus und charakterisieren jeden der in einem solchen Netzwerk auf eine Netzwerkkomponente möglichen $N+1$ regulatorischen Einflüsse durch die Schätzung eines entsprechenden Gewichts[2]. Die Struktur des Netzwerks wird damit nur implizit erlernt. Man nimmt an, daß die gegebenen Daten hier mindestens

$$N + 1 \tag{5.6}$$

verschiedene Zustandsübergänge repräsentieren müssen, um genügend Informationen für die Rekonstruktion des Netzwerks zu liefern [13]. Bei der Generierung der Zeitreihe wurde deshalb überprüft, ob sich die ersten $N+2$ gemessenen Netzwerkzustände $S(0), S(1), ..., S(N+1)$ voneinander unterscheiden. Traf dies nicht zu, wurde das Netzwerk verworfen. Zwei Netzwerkzustände $S(t_m)$ und $S(t_n)$ waren hier unterschiedlich, wenn mindestens eine Netzwerkkomponente x_i existierte, deren Werte $x_i(t_m)$ und $x_i(t_n)$, gerundet auf eine Stelle nach dem Komma, verschieden voneinander waren.

[2]Der regulatorische Einfluß des Biasfaktors wurde in die Abschätzung einbezogen.

Daten aus biologischen Experimenten unterliegen Meßfehlern. Deshalb sollten auch die generierten Simulationsdaten fehlerbehaftet sein. In Anlehnung an [13] wurde mit einem normalverteilten Fehlermodell gearbeitet. Die Expressionsrate selbst legte dabei die Varianz des ihr zugefügten Fehlers fest:

$$error_i(t) \sim N(0, rate \cdot x_i(t)) \tag{5.7}$$

Für die in dieser Arbeit durchgeführten Experimente betrug die Fehlerrate $rate$ 5%. Der Betrag des Fehlers $error_i(t)$ durfte einen bestimmten Wert $error_{i,max}(t)$ nicht überschreiten. Dieser war jeweils auf 10% des wahren Werts $x_i(t)$ festgelegt:

FALLS $(|error_i(t)| > error_{i,max}(t) = x_i(t) \cdot 0.1)$
$\rightarrow x_i^{error}(t) = x_i(t) \pm error_{i,max}(t)$
SONST
$\rightarrow x_i^{error}(t) = x_i(t) + error_i(t)$

Hierbei war auf die Einhaltung der Intervallgrenzen $(0, max_i)$ zu achten.
Für die entsprechende Diskretisierung der Simulationsdaten dienten in dieser Arbeit die Schwellwerte $\frac{max_i}{2}$ (Abbildung der gemessenen Expressionsraten auf die Zustände „0“ und „1“ für den Algorithmus *Reveal*) bzw. $\frac{max_i}{3}$ und $\frac{2 \cdot max_i}{3}$ (Abbildung der Expressionsraten auf die Zustände „-1“, „0“ und „1“ für den Algorithmus *Strukturlernen in diskreten DBN*). Die maximalen Expressionsraten der Netzwerkkomponenten wurden dafür empirisch aus den fehlerbehafteten Simulationsdaten bestimmt. Dazu ordnet man der maximalen Expressionsrate max_i einer Netzwerkkomponente x_i den kleinsten ganzzahligen Wert zu, der größer war als alle beobachteten Expressionsraten der Netzwerkkomponente.

5.1.3 Evaluierungsmaße

Die Motivation für die Arbeit mit Simulationsdaten liefert die Tatsache, daß die von den Reverse Engineering Algorithmen erzielten Ergebnisse genau evaluiert werden können. Um zu bewerten, wie gut es einem Reverse Engineering Algorithmus gelingt, die Struktur des den Daten zugrundeliegenden Netzwerks zu rekonstruieren, wird zunächst die Struktur des von ihm rekonstruierten Netzwerk mit der Struktur des zugrundeliegenden Netzwerks verglichen und die Anzahl der korrekt identifizierten regulatorischen Einflüsse $truePos$, die Anzahl der falsch identifizierten regulatorischen Einflüsse $falsePos$ sowie die Anzahl der nicht identifizierten regulatorischen Einflüsse $falseNeg$ bestimmt (vergleiche auch Abbildung 5.1):

Definition 5.1 (truePos) *Der Wert truePos beschreibt die Anzahl der von dem Reverse Engineering Algorithmus korrekt identifizierten regulatorischen Einflüsse, d.h. die Anzahl derjenigen Einflüsse, die sowohl in dem rekonstruierten als auch in dem zugrundeliegenden Netzwerk enthalten sind.*

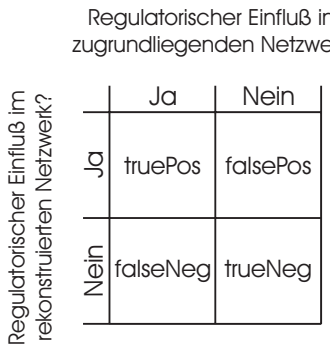


Abbildung 5.1: **Vergleich zwischen rekonstruiertem und den Expressionsdaten zugrundeliegendem Netzwerk.**

Definition 5.2 (falsePos) *Der Wert $falsePos$ zählt die falsch identifizierten regulatorischen Einflüsse, die zwar im rekonstruierten Netzwerk beschrieben sind, aber in dem zugrundeliegenden Netzwerk nicht bestehen.*

Definition 5.3 (falseNeg) *Der Wert $falseNeg$ entspricht der Anzahl der regulatorischen Einflüsse, die zwar in dem zugrundeliegenden Netzwerk existieren, die der Reverse Engineering Algorithmus aber nicht identifizieren konnte und die deshalb nicht in dem rekonstruierten Netzwerk modelliert werden.*

Auf Basis dieser Hilfsgrößen lassen sich dann die Sensitivität sowie der positiv prädiktive Wert[3] des Algorithmus bestimmen.

Definition 5.4 (Sensitivität) *Die Sensitivität gibt den prozentualen Anteil regulatorischer Einflüsse in dem den Daten zugrundeliegenden Netzwerk an, die der Reverse Engineering Algorithmus identifizieren konnte.*

$$\text{Sensitivität} = \frac{truePos}{truePos + falseNeg} \tag{5.8}$$

Definition 5.5 (positiv prädiktiver Wert) *Der positiv prädiktive Wert liefert den prozentualen Anteil der von einem Reverse Engineering Algorithmus identifizierten regulatorischen Einflüsse, die tatsächlich in dem den Daten zugrundeliegenden Netzwerk enthalten sind.*

[3] Anstelle des positiv prädiktiven Wertes könnte auch die Spezifität als weiteres Evaluierungsmaß betrachtet werden. Sie beschreibt die Wahrscheinlichkeit, daß der Reverse Engineering Algorithmus einen in dem den Daten zugrundeliegenden Netzwerk nicht existenten regulatorischen Einfluß auch als solchen einstuft. Für die Evaluierung von Reverse Engineering Algorithmen wurde hier aber der positiv prädiktive Wert gewählt, weil er für diese konkrete Anwendung im Vergleich zur Spezifität aussagekräftiger scheint.

$$positiv\ prädiktiver\ Wert = \frac{truePos}{truePos + falsePos} \tag{5.9}$$

Sie bilden wichtige Größen zur Berechnung der Güte eines rekonstruierten Netzwerks und dienen im folgenden zur Bewertung der Ergebnisse der einzelnen Algorithmen. Für die Berechnung der Güte eines rekonstruierten Netzwerks wurde also nur die Existenz identifizierter regulatorischer Einflüsse überprüft und dabei außer acht gelassen, ob auch die Qualität eines regulatorischen Einflusses (aktivierender oder inhibitorischer Einfluß) richtig festgelegt wurde.

5.2 Abhängigkeit von Netzwerkparametern, Datenumfang und Meßfehlern

In diesem Abschnitt wird zunächst analysiert, inwieweit die wichtigen Netzwerkparameter Konnektivität k und Netzwerkgröße N die Güte der Ergebnisse der einzelnen Algorithmen beeinflussen. Bezüglich der verfügbaren Daten soll dann auch die Abhängigkeit vom Datenumfang und von Meßfehlern in den Daten untersucht werden.
Die Simulationsdaten stammen aus Modellnetzwerken, die stark von der biologischen Realität abstrahieren. Die von den Algorithmen erzielten Ergebnisse lassen sich deshalb nicht direkt auf die Arbeit mit realen Expressionsdaten übertragen. Der Schwerpunkt bei den folgenden Analysen liegt deshalb auf dem grundlegenden Verhalten der Algorithmen und nicht auf den im einzelnen erzielten Werten für Sensitivität und positiv prädiktiven Wert.

Konnektivität des Netzwerks

Dieses erste Simulationsexperiment sollte die Abhängigkeit der Ergebnisse der einzelnen Reverse Engineering Algorithmen von der Konnektivität der zugrundeliegenden Netzwerke untersuchen. Es wurden für die Konnektivitäten 1, 2, 3, 4, 5 bzw. 8 jeweils 100 zufällige Netzwerke der Netzwerkgröße 20 erzeugt und anschließend fehlerbehaftete Simulationsdaten generiert.
Bei derzeit verfügbaren Expressionsdaten handelt es sich neben stabilen Zustandsdaten meist um Zeitreihen. Deshalb ist die Arbeit mit Zeitreihen realistischer als die Arbeit mit Zustandsübergangsdaten, auch wenn eine Zeitreihe lediglich eine ausgewählte Trajektorie des Netzwerks beschreibt und damit weniger Informationen über das dynamische Verhalten des Netzwerks liefert als die unabhängigen Zustandsübergänge der Zustandsübergangsdaten. Wie in Abschnitt 5.1.2 zur Generierung der Simulationsdaten erläutert, orientiert sich diese Arbeit an dem Additiven Regulationsmodell, um die Mindestanzahl benötigter Zustandsübergänge abzuschätzen. Für ein Netzwerk der Größe N sollten hier die gegebenen Daten idealerweise mindestens $N + 1$ verschiedene Zustandsübergänge charakterisieren. Die

Algorithmen bekamen deshalb bei diesem Experiment für jedes zu untersuchende Netzwerk eine – gegebenenfalls entsprechend diskretisierte – Zeitreihe übergeben, die 22 aufeinanderfolgende, unterschiedliche Netzwerkzustände beschrieb. Es ist anzunehmen, daß eine Zeitreihe mit diesem Datenumfang für die Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* aufgrund der Diskretisierung nicht genügend Informationen enthält. Ziel soll es hier aber nicht sein, den benötigten Datenumfang für die einzelnen Algorithmen zu untersuchen, sondern zu analysieren, welche Ergebnisse mit derzeit verfügbaren Daten erzielt werden können. Bereits das hier vorausgesetzte Verhältnis von $1:1$ zwischen dem Datenumfang und der Anzahl der Netzwerkkomponenten ist bezüglich der derzeit bereitstehenden realen Expressionsdaten relativ optimistisch.
Im Gegensatz zu allen anderen Algorithmen bekommt die *Adjazenzlisten-Konstruktion* stabile Zustandsdaten übergeben, die die Auswirkungen aller, in einem Netzwerk der Größe N möglichen N Manipulationen einer einzelnen Netzwerkkomponente präsentieren.

Abbildung 5.2 zeigt die Ergebnisse, die sich bei der Anwendung aller in Tabelle 5.1 zusammengefaßten Reverse Engineering Methoden auf die Simulationsdaten ergaben. Es ist sowohl die Abhängigkeit der Sensitivität von der Konnektivität als auch die Abhängigkeit des positiv prädiktiven Wertes von der Konnektivität in Diagrammen graphisch dargestellt.
Offensichtlich führt eine höhere Konnektivität zu einer kleineren Sensitivität, aber auch zu einem größeren positiv prädiktiven Wert. Dabei zeigt sich zwischen dem positiv prädiktiven Wert und der Konnektivität für fast alle Reverse Engineering Methoden ein annähernd linearer Zusammenhang, während die Sensitivität exponentiell mit steigender Konnektivität fällt.
Der Zusammenhang zwischen dem positiv prädiktiven Wert und der Konnektivität läßt sich durch die folgende Tatsache erklären: In einem Netzwerk der Größe 20 sind 400 regulatorische Einflüsse zwischen den Netzwerkkomponenten möglich. Der Anteil tatsächlich existierender regulatorischer Einflüsse an dieser Anzahl potentieller Einflüsse steigt mit zunehmender Konnektivität. Existieren bei einer Konnektivität von 1 lediglich 20 von 400 möglichen regulatorischen Einflüssen, so beträgt der Anteil bei einer Konnektivität von 4 schon 80 von 400 und bei einer Konnektivität von 8 sogar 160 von 400. Damit steigt bei zunehmender Konnektivität auch die Wahrscheinlichkeit, daß ein potentieller regulatorischer Einfluß tatsächlich existiert, worauf sich die Zunahme des positiv prädiktiven Wertes zurückführen läßt.
Bei einer Konnektivität größer 1 wirken mehrere Netzwerkkomponenten additiv auf eine Netzwerkkomponente x_i ein. Der Anteil einer einzelnen dieser Netzwerkkomponenten am gesamten regulatorischen Einfluß auf x_i sinkt mit steigender Konnektivität und kann auch sehr gering ausfallen. Er ist dann, vor allem mit fehlerbehafteten Daten, schwerer nachweisbar. Die Sensitivität sinkt. Besonders gravierende Auswirkungen hat dies bei den Reverse Engineering Algorithmen, die mit diskreten

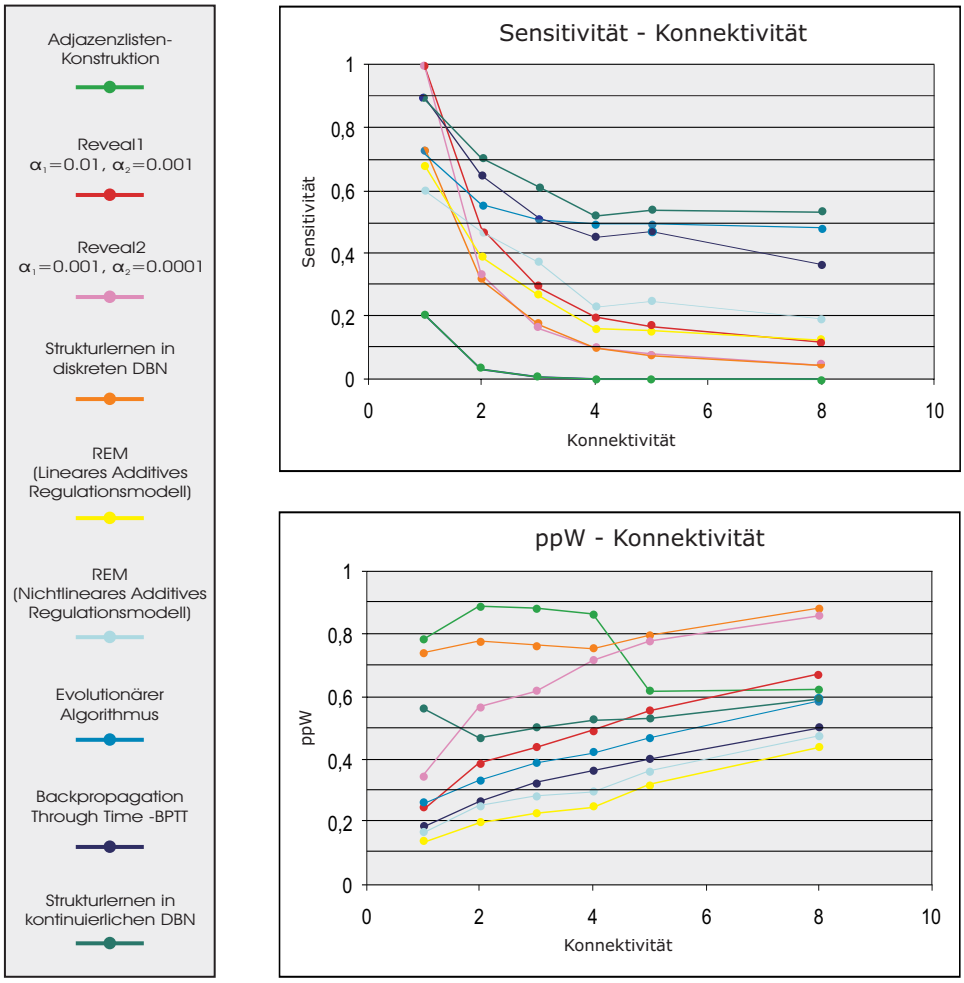


Abbildung 5.2: **Abhängigkeit von der Konnektivität k der zugrundeliegenden Netzwerke.** Jeder Datenpunkt in den Diagrammen entspricht dabei den über jeweils alle 100 betrachteten Netzwerke gemittelten Werten für Sensitivität bzw. positiv prädiktiven Wert. Alle betrachteten Netzwerke hatten eine konstante Netzwerkgröße N von 20. Den Algorithmen wurde jeweils eine Zeitreihe der Länge $N + 1$ übergeben.

Expressionsraten arbeiten (*Reveal*, *Strukturlernen in diskreten DBN*). Bei höheren Konnektivitäten ist es diesen Algorithmen kaum noch möglich, regulatorische Zusammenhänge zu erkennen. Dies spiegelt sich in einer kleinen Sensitivität und einem recht hohen positiv prädiktiven Wert wider. Ihre Sensitivität fällt wesentlich steiler ab als die der anderen Reverse Engineering Algorithmen. Zeigt beispielsweise der Algorithmus *Reveal* bei einer Konnektivität von 1 eine höhere Sensitivität als alle anderen Algorithmen, so fällt sie bei einer Konnektivität von 2 schon unter die der meisten anderen ab. Folgender Sachverhalt könnte diese Beobachtung erklären: Obwohl die Abschätzungen 5.4 und 5.5 über den von ihnen benötigten Datenumfang

hier nicht verwendet werden können, so lassen sie doch vermuten, daß dieser von der Konnektivität k abhängig ist. Der Unterschied zwischen der für die Rekonstruktion benötigten Information und der in den gegebenen Daten bereitgestellten Information wird deshalb mit steigender Konnektivität immer größer. Den Algorithmen stehen verhältnismäßig weniger Informationen zur Verfügung, wodurch sich die Identifizierung additiver regulatorischer Einflüsse zusätzlich erschwert. Die Abschätzung 5.6 von $N + 1$ benötigten, unabhängigen Zustandsübergängen für Reverse Engineering Algorithmen, die auf dem Additiven Regulationsmodell basieren, ist dagegen beispielsweise unabhängig von der Konnektivität k, und so fällt die Sensitivität dieser Algorithmen nicht so stark ab.

Die schlechten Ergebnisse der *Adjazenzlisten-Konstruktion* sind vor allem auf die Zyklen in den Netzwerken zurückzuführen. Wie in Abschnitt 3.1.1 beschrieben, werden alle Netzwerkkomponenten eines Zyklus vor der Konstruktion der Adjazenzliste zusammengefaßt. Ein identifizierter regulatorischer Einfluß von oder zu einer solchen Zykluskomponente wurde hier bei der Auswertung der Ergebnisse nicht berücksichtigt, da nicht eindeutig zugeordnet werden konnte, von welcher Netzwerkkomponente im Zyklus der Einfluß ausgeht, bzw. auf welche er einwirkt. Damit kann der Algorithmus nur regulatorische Einflüsse zwischen solchen Netzwerkkomponenten finden, die in keinem Zyklus involviert sind. Die betrachteten Modellnetzwerke besaßen bei einer Konnektivität von 1 im Schnitt 5.38 solcher nachweisbaren regulatorischen Einflüsse. Steigt die Konnektivität, sinkt diese Anzahl weiter – schon bei einer Konnektivität von 2 betrug sie nur noch 1.12 und bei der Konnektivität 3 sogar nur noch 0.09. Es ist dann dem Algorithmus kaum noch möglich, irgendeinen Einfluß nachzuweisen, und die konstruierte Adjazenzliste enthält bei hohen Konnektivitäten nur sehr wenige bzw. gar keine Einträge; die Sensitivität erreicht den Wert 0.
Bei der Anwendung eines statistischen Tests zur Generierung der Erreichbarkeitsliste können auch Fehler begangen werden. Deshalb müssen nicht alle von dem Algorithmus identifizierten Einflüsse korrekt sein. Die Identifizierung eines falschen regulatorischen Einflusses hat bei einer kleinen Menge an – korrekt oder falsch – identifizierten Einflüssen große Auswirkungen auf den prozentualen Anteil der korrekt identifizierten regulatorischen Einflüsse. Dies erklärt, weshalb der positiv prädiktive Wert bei der *Adjazenzlisten-Konstruktion*, im Gegensatz zu allen anderen Algorithmen, mit steigender Konnektivität sinkt.
An dieser Stelle sollte man berücksichtigen, daß die hier untersuchten Konnektivitäten der betrachteten Modellnetzwerke bezüglich realer Genregulationsnetzwerke zwar durchaus realistisch sind, dafür aber eine sehr kleine Netzwerkgröße angenommen wurde. Reale Genregulationsnetzwerke sind in der Regel sehr viel größer. Bei gleicher Konnektivität sinkt mit steigender Netzwerkgröße für eine beliebige Netzwerkkomponente die Wahrscheinlichkeit, in einem Zyklus involviert zu sein. Die Anzahl nachweisbarer regulatorischer Einflüsse steigt. Deshalb ist anzunehmen, daß

dieser Algorithmus bei der Arbeit mit realen Expressionsdaten bessere Ergebnisse liefern kann, falls die strukturellen Eigenschaften realer Genregulationsnetzwerke nicht gerade die Bildung von Zyklen unterstützen.

Weiterhin verdeutlichen die Ergebnisse den Einfluß der Signifikanzniveaus α_1 und α_2 für den Algorithmus *Reveal*. Wie in Abschnitt 3.2.1 erläutert, beschreibt das Signifikanzniveau die Irrtumswahrscheinlichkeit 1. Art, eine richtige Nullhypothese abzulehnen und fälschlicherweise Elemente in die Elternmenge einer Netzwerkkomponente aufzunehmen. Sie hat Einfluß auf die Irrtumswahrscheinlichkeit 2. Art β, eine falsche Nullhypothese nicht abzulehnen und fälschlicherweise Elternelemente nicht in die Elternmenge einer Netzwerkkomponente aufzunehmen. Eine kleinere Irrtumswahrscheinlichkeit 1. Art vergrößert die Irrtumswahrscheinlichkeit 2. Art. Deshalb erlangt *Reveal1* in diesem Experiment eine höhere Sensitivität, aber einen schlechteren positiv prädiktiven Wert als *Reveal2*, der mit kleineren Signifikanzniveaus arbeitet.

Über den Vergleich der Algorithmen läßt sich folgendes sagen: Der Algorithmus *Strukturlernen in kontinuierlichen DBN* liefert bessere Ergebnisse als der *evolutionäre Algorithmus* sowie die Algorithmen *REM* (nichtlinearer Ansatz) und *BPTT*. Diese Überlegenheit resultiert aus der Tatsache, daß er die Rekonstruktion des Netzwerks durch ein explizites Erlernen der Struktur unterstützt, während die anderen Algorithmen die Struktur nur implizit über die Schätzung der Gewichtsmatrix bestimmen. Durch den wahrscheinlichkeitstheoretischen Hintergrund des kontinuierlichen DBN kann er Fehler und Inkonsistenzen in den Daten zudem besser berücksichtigen.
In der Gruppe der mit dem nichtlinearen Additiven Regulationsmodell arbeitenden Algorithmen liefert der Algorithmus *REM* vor allem bezüglich der Sensitivität schlechtere Ergebnisse als der Algorithmus *BPTT* und der *evolutionäre Algorithmus*. Erwartungsgemäß weisen Reverse Engineering Algorithmen, die auf dem nichtlinearen Additiven Regulationsmodell (*REM, BPTT, evolutionärer Algorithmus*) bzw. auf dem an diesem Modell orientierten kontinuierlichen DBN (*Strukturlernen in kontinuierlichen DBN*) basieren, eine höhere Sensitivität und einen kleineren positiv prädiktiven Wert auf als alle anderen Reverse Engineering Algorithmen. Sie können regulatorische Einflüsse besser nachweisen, begehen dabei aber auch mehr Fehler. Hier ist zu berücksichtigen, daß das nichtlineare Additive Regulationsmodell auch als Basis für die Generierung der Simulationsdaten diente. Für diese Algorithmen kommen deshalb viele ihrer modellbedingten Limitationen nicht zum Tragen, und ihre Ergebnisse können nicht mit den Ergebnissen der übrigen Algorithmen verglichen werden. Denn während sie lediglich mit Meßfehlern in den Daten und mit Schwierigkeiten bei der empirischen Bestimmung der maximalen Expressionsraten[4] umgehen müssen, kommen für die anderen Algorithmen zusätzlich erschwe-

[4]Siehe Abschnitt *Limitationen* in 3.4.1.

rend vereinfachte bzw. falsche Modellannahmen hinzu. Für den Algorithmus *REM* zur Anpassung eines linearen Additiven Regulationsmodells ist das im wesentlichen die fälschlicherweise angenommene Linearität der regulatorischen Interaktionen. Für die auf Booleschen Netzwerken und diskreten DBN basierenden Algorithmen hingegen erschwert die vereinfachende Betrachtung diskreter Expressionsraten die Rekonstruktion der zugrundeliegenden Netzwerke erheblich.
Aufgrund der vereinfachten und falschen Modellannahmen liefern die übrigen Algorithmen gerade bei großer Konnektivität bezüglich der Sensitivität nur sehr unbefriedigende Ergebnisse. Während die Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* jedoch einen relativ hohen positiv prädiktiven Wert erzielen, muß für die mit dem linearen Additiven Regulationsmodell arbeitende Variante des Algorithmus *REM* außerdem ein unbefriedigender positiv prädiktiver Wert beobachtet werden. Bemerkenswert ist die Tatsache, daß es dem Algorithmus *Reveal* bei einer Konnektivität von 1 trotz seiner Benachteiligung aufgrund vereinfachter Modellannahmen gelingt, insgesamt bessere Ergebnisse zu erzielen als alle anderen betrachteten Algorithmen.

Größe des Netzwerks

In diesem Simulationsexperiment variierte die Netzwerkgröße N der erzeugten Netzwerke, um den Einfluß dieser Größe auf die Ergebnisse der Reverse Engineering Algorithmen zu untersuchen. Es wurden für jede der Netzwerkgrößen 10, 20, 30, 40 und 50 jeweils 50 zufällige Netzwerke erzeugt. Alle Netzwerke hatten eine konstante Konnektivität von 3. Eine Ausnahme bildet die *Adjazenzlisten-Konstruktion*. Das vorangegangene Experiment hat verdeutlicht, daß dieser Algorithmus bei einer Konnektivität von 3 kaum noch regulatorische Zusammenhänge identifizieren kann. Da der Einfluß der Netzwerkgröße aber auch für diesen Algorithmus untersucht werden sollte, wurden für ihn entsprechende Netzwerke mit einer Konnektivität von 1 erzeugt.
Als Eingabe bekamen die Reverse Engineering Algorithmen für jedes Netzwerk eine entsprechende Zeitreihe der jeweiligen Länge $N+1$ bzw. stabile Zustandsübergangsdaten, die die Ergebnisse aus N verschiedenen Manipulationen eines einzelnen Gens präsentieren, übergeben.

Die Ergebnisse dieses Simulationsexperiments findet man in Abbildung 5.3. Wieder sind die entsprechenden Zusammenhänge zwischen Sensitivität bzw. positiv prädiktivem Wert und der Netzwerkgröße N in Diagrammen graphisch dargestellt.
Für den Zusammenhang zwischen Sensitivität und Netzwerkgröße fällt folgendes auf: Die Ergebnisse der Algorithmen *REM* (linearer und nichtlinearer Ansatz) und *BPTT* sowie des *evolutionären Algorithmus* zeigen bei zunehmender Netzwerkgröße eine kleinere Sensitivität. Eine Erklärung für diese Beobachtung könnte der Sachverhalt liefern, daß die Anzahl der prinzipiell möglichen regulatorischen Einflüsse steigt. Da diese Algorithmen mit einem vollständig verknüpften Netzwerk arbeiten,

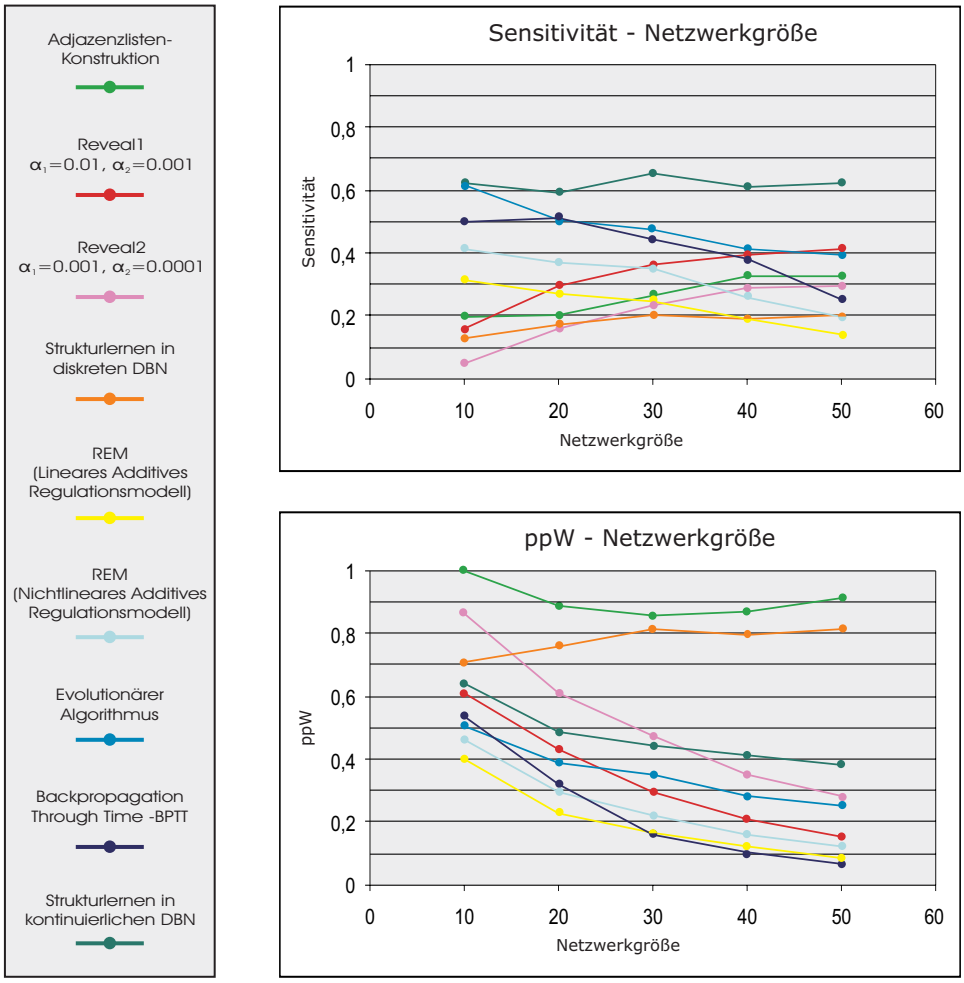


Abbildung 5.3: **Abhängigkeit von der Netzwerkgröße N der zugrundeliegenden Netzwerke.** Jeder Datenpunkt in den Diagrammen entspricht den gemittelten Werten für Sensitivität bzw. positiv prädiktiven Wert über jeweils alle 50 betrachteten Netzwerke. Die Konnektivität der Netzwerke betrug konstant den Wert 3. Eine Ausnahme bildet die *Adjazenzlisten-Konstruktion*, deren betrachteten Netzwerke eine Konnektivität von 1 aufwiesen. Den Algorithmen wurde jeweils eine Zeitreihe der Länge $N + 1$ übergeben.

müssen sie in größeren Netzwerken auch zunehmend mehr regulatorische Einflüsse durch die Schätzung entsprechender Gewichte charakterisieren und diese geeignet in „null“ und „nicht-null“ klassifizieren. Dies erschwert die korrekte Charakterisierung und Klassifizierung der Einflüsse. Die Sensitivität sinkt. Für diese Erklärung spricht auch die Tatsache, daß die Abnahme der Sensitivität nicht für den Algorithmus *Strukturlernen in kontinuierlichen DBN* beobachtet werden kann. Das kontinuierliche DBN, auf dem dieser Algorithmus basiert, ist am nichtlinearen Additiven Regulationsmodell orientiert, und so muß auch dieser Algorithmus geeignete Schätzer für

die einzelnen Gewichte w_{ij} spezifizieren. Im Gegensatz zu den direkt auf dem Additiven Regulationsmodell basierenden Algorithmen verbindet er aber die Schätzung der Gewichte mit einer expliziten Suche nach einem guten Schätzer für die Struktur des Netzwerks. Die Sensitivität seiner Ergebnisse ist deshalb unabhängig von der Netzwerkgröße.
Die beschriebene Zunahme potentieller regulatorischer Einflüsse bei zunehmender Netzwerkgröße liefert auch eine Erklärung für die steigende Sensitivität der Resultate der *Adjazenzlisten-Konstruktion*. Der Anteil der tatsächlich auf eine Netzwerkkomponente einwirkenden regulatorischen Einflüsse an der Menge prinzipiell möglicher regulatorischer Einflüsse nimmt ab; die Wahrscheinlichkeit, daß eine Netzwerkkomponente in einem Zyklus involviert ist, sinkt. Damit nimmt auch die Anzahl der für den Algorithmus nachweisbaren regulatorischen Interaktionen in den betrachteten Modellnetzwerken zu. Beträgt diese Anzahl nachweisbarer Einflüsse in den Netzwerken der Größe 10 im Schnitt nur 1.98, so steigt sie bei einer Größe von 30 auf 11.88 und beträgt für $N = 50$ sogar 25.5. Das Verhältnis zwischen nachweisbaren und existierenden regulatorischen Interaktionen steigt also, und die Sensitivität nimmt zu.
Die mit diskreten Expressionsraten arbeitenden Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* können ebenfalls die Sensitivität ihrer Ergebnisse bei zunehmender Netzwerkgröße steigern. Mit der Netzwerkgröße N nahm in diesem Experiment der Umfang der verfügbaren Simulationsdaten und damit auch die bereitgestellte Information über das dynamische Verhalten der Netzwerke linear zu. Auch wenn die existierenden Abschätzungen 5.4 und 5.5 über den benötigten Datenumfang des *Reveal* Algorithmus bei diesen Simulationen nicht verwendet werden konnten, so lassen sie doch die Vermutung zu, daß dieser lediglich vom Logarithmus der Netzwerkgröße $\log(N)$ abhängig ist. Dem Algorithmus wird bei steigender Netzwerkgröße deshalb verhältnismäßig mehr Information für die Rekonstruktion des Netzwerks bereitgestellt; die Sensitivität seiner Ergebnisse steigt. Diese Erklärung begründet vermutlich auch den Anstieg der Sensitivität des Algorithmus *Strukturlernen in diskreten DBN*.

Für den positiv prädiktiven Wert kann mit zunehmender Netzwerkgröße ein exponentielles Abfallen beobachten werden. Neben den korrekt identifizierten regulatorischen Einflüssen liefern die meisten Algorithmen also bei steigender Netzwerkgröße verhältnismäßig immer mehr falsch identifizierte regulatorische Interaktionen. Auch diese Beobachtung ist durch die mit steigender Netzwerkgröße zunehmende Anzahl an potentiellen regulatorischen Einflüssen und dem damit – ebenfalls exponentiell – abfallenden Anteil von tatsächlich existierenden regulatorischen Einflüssen an der Menge dieser prinzipiell möglichen regulatorischen Interaktionen zu erklären[5]: Die

[5]Bei einer Konnektivität von 3 und einer Netzwerkgröße 10 beträgt das Verhältnis zwischen tatsächlich existierenden und potentiellen regulatorischen Einflüssen 30:100 (30%), bei N=20 60:400 (15%), bei N=30 90:900 (10%), bei N=40 120:1600 (7.5%) und bei N=50 150:2500 (6%).

Wahrscheinlichkeit, daß ein möglicher regulatorischer Einfluß im zugrundeliegenden Netzwerk tatsächlich existiert, nimmt exponentiell mit größerem N ab und führt zu einem entsprechend abfallenden positiv prädiktiven Wert.
Eine Ausnahme bildet hier der Algorithmus *Strukturlernen in diskreten DBN*. Er zeigt bei steigender Netzwerkgröße sogar einen leichten Anstieg des ohnehin schon recht hohen positiv prädiktiven Wertes. Die Zunahme des verfügbaren Datenumfangs ermöglicht es ihm noch besser, zwischen korrekten und falschen regulatorischen Einflüssen zu unterscheiden.

Wieder ist der beschriebene Einfluß der Signifikanzniveaus α_1 und α_2 für den Algorithmus *Reveal* zu erkennen. *Reveal1* arbeitet mit größeren Signifikanzniveaus als *Reveal2*, begeht deshalb häufiger einen Fehler 1. Art und liefert einen kleineren positiv prädiktiven Wert. Gleichzeitig ist die Irrtumswahrscheinlichkeit β für einen Fehler 2. Art geringer, und *Reveal1* kann eine höhere Sensitivität als *Reveal2* erzielen.

Auch dieses Experiment bestätigt eine deutliche Überlegenheit des Algorithmus *Strukturlernen in kontinuierlichen DBN* gegenüber den Algorithmen, die direkt mit dem nichtlinearen Additiven Regulationsmodell arbeiten.
Der Algorithmus *REM* liefert in der Gruppe der auf dem nichtlinearen Additiven Regulationsmodell basierenden Algorithmen ebenfalls wiederum die schlechteren Ergebnisse. Der *evolutionäre Algorithmus* kann hier vor allem bezüglich des positiv prädiktiven Wertes gegen den Algorithmus BTPP überzeugen.
Wie bereits im vorangegangenen Experiment begründet, können die Ergebnisse der Algorithmen *REM* (nichtlinearer Ansatz), *BPTT*, *Strukturlernen in kontinuierlichen DBN* sowie des *evolutionären Algorithmus* nicht mit den Ergebnissen der übrigen Algorithmen verglichen werden. Bei den mit diskreten Netzwerkmodellen arbeitenden Algorithmen zeichnet sich der Algorithmus *Strukturlernen in diskreten DBN* durch einen sehr hohen positiv prädiktiven Wert aus. Allerdings gelingt es dem *Reveal* Algorithmus besser, die Sensitivität seiner Ergebnisse mit zunehmender Netzwerkgröße zu steigern. Selbst bei großen Netzwerkgrößen reicht die mit den Daten bereitgestellte Information für den Algorithmus *Strukturlernen in diskreten DBN* kaum aus, um regulatorische Einflüsse zu identifizieren; er kann nur eine recht kleine Sensitivität vorweisen. Dem Algorithmus *REM* (linearer Ansatz) ist es vor allem aufgrund eines recht kleinen positiv prädiktiven Wertes nicht möglich, sich gegen die beiden mit diskreten Expressionsraten arbeitenden Algorithmen durchzusetzen.
Die Ergebnisse für den Algorithmus *Adjazenzlisten-Konstruktion* stützen die im vorangegangenen Experiment beschriebene Annahme, daß dieser Algorithmus bei der Arbeit mit realen Expressionsdaten aufgrund einer sehr viel höheren Netzwerkgröße realer genetischer Netzwerke durchaus brauchbare Ergebnisse liefern kann. Auch dieser Algorithmus liefert unabhängig von der Netzwerkgröße einen sehr hohen positiv prädiktiven Wert. Seine Ergebnisse können nicht in den Vergleich der Algorithmen einbezogen werden, da die von ihm betrachteten Modellnetzwerke lediglich eine Kon-

nektivität von 1 aufwiesen, während alle anderen Algorithmen mit Netzwerken der Konnektivität 3 arbeiteten.

Umfang der gegebenen Daten – Zeitreihe

Um die Abhängigkeit der Reverse Engineering Methoden von der Anzahl gegebener Zustandsübergänge zu untersuchen, wurden 100 zufällige Modellnetzwerke erzeugt. Sie bestanden aus je 20 Netzwerkkomponenten und wiesen eine konstante Konnektivität von 3 auf. Anschließend überprüfte dieses Simulationsexperiment, inwieweit es den einzelnen Algorithmen gelang, diese 100 Modellnetzwerke mit einer in der jeweiligen fehlerbehafteten Zeitreihe zur Verfügung gestellten Anzahl M beobachteter Zustandsübergänge von 6, 11, 16, 21, 35, 50, 75 bzw. 100 zu rekonstruieren. Nicht betrachtet wurde hier die *Adjazenzlisten-Konstruktion*. Dieser Algorithmus arbeitet nicht mit einer Zeitreihe, sondern mit stabilen Zustandsdaten, die die Ergebnisse aller N möglichen Manipulationen einer einzelnen Netzwerkkomponente charakterisieren müssen.

Die aus dieser Untersuchung resultierenden Ergebnisse sind in Abbildung 5.4 zu finden. Das obere Diagramm beschreibt hier die Abhängigkeit der über alle 100 Netzwerke gemittelten Sensitivität von der Anzahl verfügbarer Zustandsübergänge. Analog stellt das untere Diagramm die Abhängigkeit des gemittelten positiv prädiktiven Wertes vom Datenumfang dar.
Wie zu erwarten, lieferten die Algorithmen – bis auf einige Ausnahmen – mit zunehmendem Datenumfang M bessere Ergebnisse; sie können sowohl die Sensitivität als auch den positiv prädiktiven Wert steigern. Der Anstieg erfolgt dabei im allgemeinen logarithmisch. Speziell für die Algorithmen, die auf dem nichtlinearen Additiven Regulationsmodell bzw. auf dem an diesem Modell orientierten kontinuierlichen DBN basieren, liegt dies zum einen daran, daß das von diesen Algorithmen zur Bestimmung des Schätzers der Gewichtsmatrix W zu lösende Gleichungssystem (siehe Gleichung 3.34) unterbestimmt ist, wenn weniger als $N + 1$ beobachtete Zustandsübergänge vorliegen. Zusätzlich beobachtete Zustandsübergänge bis zu einem Datenumfang von $N + 1$, in diesem speziellen Fall also 21, liefern deshalb besonders viel Information über das dynamische Verhalten des Systems, erlauben so eine wesentlich genauere Schätzung und Klassifizierung der Gewichte w_{ij}, und es ist ein relativ hoher Anstieg der beiden Größen Sensitivität und positiv prädiktiver Wert zu verzeichen. Eine weitere Zunahme des Datenumfangs kann, vor allem bei fehlerbehafteten Daten, weitere Verbesserungen bringen, die aber geringer ausfallen. Der eigentliche Hauptgrund dafür, daß sich die Ergebnisse dieser und auch der übrigen Algorithmen gerade bei größerem Datenumfang kaum noch verbessern, ist aber auf die Arbeit mit Zeitreihen zurückzuführen. Bei der Generierung dieser wurde zwar sichergestellt, daß die Netzwerke bis zu dem Zeitpunkt $N + 1$ nicht in einen Attraktor gelangten und die jeweils zugehörige Zeitreihe somit $N + 1$ unterschiedliche Systemzustände repräsentiert. Das Verhalten der Netzwerke zu einem späteren Zeit-

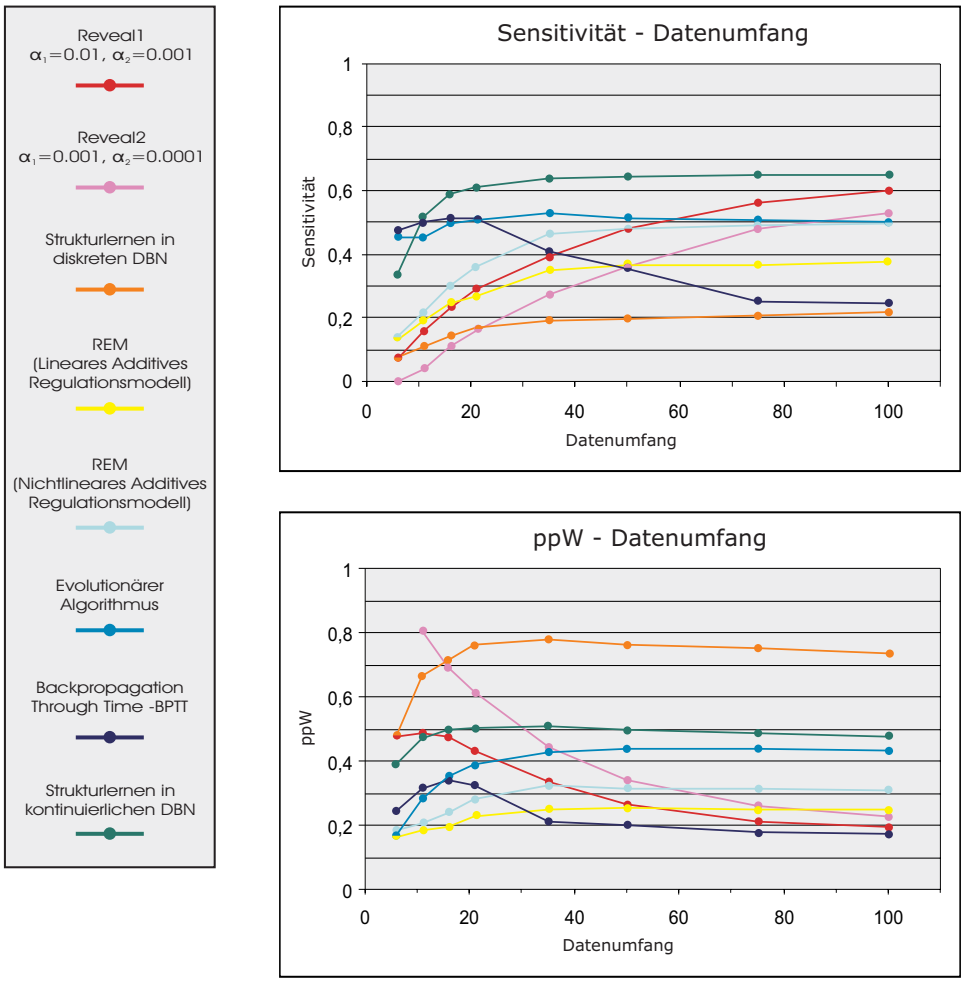


Abbildung 5.4: **Abhängigkeit vom Umfang M der gegebenen Daten – Zeitreihe.** Jeder Datenpunkt in den Diagrammen entspricht den gemittelten Werten für Sensitivität bzw. positiv prädiktiven Wert über jeweils alle 100 betrachteten Netzwerke. Alle Netzwerke bestanden aus 20 Netzwerkkomponenten; ihre Konnektivität betrug konstant den Wert 3.

punkt wurde aber nicht überprüft. Es ist jedoch sehr wahrscheinlich, daß viele von ihnen noch vor einer Anzahl von 100 Zeitschritten in einen Attraktor gelangen. Der Verlauf der Kurven läßt vermuten, daß die Netzwerke im Schnitt bereits nach 35 Zeitschritten einen Attraktor erreicht hatten. Ab diesem Zeitpunkt werden nur noch die Attraktorzustände beobachtet und das Verhalten des Systems im Attraktor wiederholt protokolliert. Eine weitere Betrachtung des Systems liefert deshalb keine neuen Systemzustände und somit auch keine neuen Informationen über sein dynamisches Verhalten. Die wiederholte Beobachtung bestimmter Systemzustände im Attraktor eines Netzwerks kann sich sogar negativ auf seine Rekonstruktion auswirken. Für den Algorithmus *Strukturlernen in DBN* – sowohl bei der Arbeit mit

einem kontinuierlichen DBN als auch bei der Verwendung eines diskreten DBN – ist ab einem Datenumfang von 35 ein leichtes Abfallen des positiv prädiktiven Wertes zu beobachten; im Falle des *evolutionären Algorithmus* zeigt sich ein ähnliches Verhalten bezüglich der Sensitivität.

Ein auffallendes Verhalten weist der Algorithmus *Reveal* auf. Es gelingt ihm bei steigendem Datenumfang wesentlich besser, die im zugrundeliegenden Netzwerk existierenden regulatorischen Einflüsse zu identifizieren als den anderen Algorithmen. Selbst bei einem Datenumfang ab 35 Zustandsübergängen nimmt die Sensitivität weiter zu. Wie eben erklärt, erreichen die betrachteten Netzwerke im Durchschnitt bereits zu einem frühen Zeitpunkt einen Attraktor und eine weitere Beobachtung des Systems liefert wiederholend bestimmte Attraktorzustände des Systems. Im Gegensatz zu allen anderen Algorithmen ist *Reveal* in der Lage, auch aus diesen Beobachtungen Information zu gewinnen und die Sensitivität weiter zu steigern.
Allerdings werden durch die wiederholte Beobachtung bestimmter Systemzustände auch Zusammenhänge vorgetäuscht, die eigentlich nicht existieren. Dies ist ein Grund dafür, daß der Algorithmus mit zunehmendem Datenumfang immer mehr falsch positive regulatorische Zusammenhänge identifiziert und der positiv prädiktive Wert erheblich sinkt.
Ein anderer Grund für dieses beobachtete Abfallen des positiv prädiktiven Wertes ist auf eine grundlegende Eigenschaft des Algorithmus zurückzuführen: Die in dieser Arbeit verwendete Implementation von *Reveal* ist eine Erweiterung des ursprünglichen *Reveal* Algorithmus, die von [42] an die Arbeit mit fehlerbehafteten Daten angepaßt wurde[6]. Zur Identifizierung von regulatorischen Einflüssen arbeitet diese Variante mit statistischen Tests. Mit zunehmendem Datenumfang lassen sich existierende regulatorische Einflüsse besser nachweisen. Der erste Test, der überprüft, ob eine Menge X von Inputelementen Information auf ein Outputelement überträgt, fällt zunehmend öfter positiv aus. Dies bewirkt eine häufigere Ausführung des zweiten Tests, der im Anschluß an den ersten Test für jedes Inputelement einer positiv getesteten Menge X untersucht, ob es tatsächlich an der Informationsübertragung beteiligt ist. Bei jeder einzelnen Ausführung dieses zweiten Tests begeht der Algorithmus mit einer Irrtumswahrscheinlichkeit α_2 einen Fehler 1. Art, lehnt eine richtige Nullhypothese ab und identifiziert fälschlicherweise einen regulatorischen Einfluß des getesteten Inputelements auf das entsprechende Outputelement. Die Irrtumswahrscheinlichkeit $\alpha_{2,gesamt}$ insgesamt, bei der mehrfachen Ausführung dieses Tests einen solchen Fehler 1. Art zu begehen, vergrößert sich mit steigender Anzahl der Ausführungen; der positiv prädiktive Wert sinkt[7].

Speziell für den *evolutionären Algorithmus* fällt auf, daß er bezüglich der Sensitivität weniger vom Datenumfang und in Bezug auf den positiv prädiktiven Wert mehr vom Datenumfang abhängt als die anderen beiden mit dem nichtlinearen Additiven Regu-

[6]Siehe Abschnitt *Der Algorithmus* in 3.2.1.
[7]Siehe Abschnitt *Limitationen* in 3.2.1.

lationsmodell arbeitenden Algorithmen. Dieses Ergebnis kann mit Hilfe der von ihm verwendeten Fitneßfunktion (siehe Gleichung 3.38) begründet werden: Ein größerer Datenumfang führt auch zu einem größeren Unterschied zwischen der Fitneß zweier qualitativ unterschiedlicher Individuen. Mit steigendem Datenumfang wird der Selektionsdruck in der Population also größer, und gute Schätzer der Gewichte w_{ij} können sich besser gegen schlechte Schätzer durchsetzen. Zusätzliche Informationen helfen damit hier primär, schlechte Schätzer zu eliminieren und so den positiv prädiktiven Wert zu steigern. Nur als Folge davon gelingt es dem Algorithmus dann bei steigendem Datenumfang besser, aus den sich durchsetzenden guten Schätzern neue, noch bessere Schätzer zu bilden und die Sensitivität zu steigern.

Auch für den *BPTT* Algorithmus ist anfänglich eine Zunahme der Sensitivität und des positiv prädiktiven Wertes zu verzeichnen. Schon ab einem Datenumfang von 21 gegebenen Zustandsübergängen fallen beide Größen – im Gegensatz zu den anderen, an dem Additiven Regulationsmodell orientierten Algorithmen – aber erheblich ab. Die zu einem Netzwerk gegebene Zeitreihe beschreibt eine bestimmte Trajektorie des Systems. Für jeden beobachteten Zeitpunkt wird die dem Algorithmus zugrundeliegende Feedforward-Netzwerkarchitektur um eine Schicht erweitert. Aufgabe des *BPTT* Algorithmus ist es, die Gewichte dieses neuronalen Netzwerks iterativ anzupassen und ihm damit das gegebene zeitliche Verhalten zu erlernen. Die schlechteren Ergebnisse bei zunehmendem Datenumfang können dadurch erklärt werden, daß eine steigende Länge der gegebenen Trajektorie und damit eine steigende Anzahl an Netzwerkschichten mehr Iterationen erfordert, um das neuronale Netzwerk entsprechend gut anzupassen. Bei der speziellen Anwendung des *BPTT* Algorithmus in dieser Arbeit wurde nach konstant 2000 Iterationen der Lernprozeß beendet. Kürzere Trajektorien können in dieser Spanne besser erlernt werden.
Sicherlich üben auch hier die beschriebenen Probleme, die durch die wiederholte Beobachtung der Attraktorzustände in den Zeitreihen entstehen, einen negativen Einfluß auf den Reverse Engineering Prozeß aus.

Die Erkenntnisse aus den vorangegangenen Experimenten über den Vergleich der Algorithmen konnten in diesem Experiment im großen und ganzen bestätigt werden: Der Algorithmus *Strukturlernen in kontinuierlichen DBN* erzielte auch hier bessere Ergebnisse als die mit dem Additiven Regulationsmodell arbeitenden Algorithmen. In der Gruppe dieser Algorithmen lieferten zumindest bei kleinem Datenumfang der Algorithmus *BPTT* und der *evolutionäre Algorithmus* bessere Ergebnisse als der Algorithmus *REM* (nichtlinearer Ansatz). Zu berücksichtigen sind hier allerdings die Probleme, die für den Algorithmus *BPTT* bei großem Datenumfang entstehen.
Die mit diskreten Expressionsraten arbeitenden Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* liefern beide nur unbefriedigende Ergebnisse. Während *Reveal* bei großem Datenumfang zwar viele regulatorische Einflüsse identifizieren kann, liefert er nur einen sehr kleinen positiv prädiktiven Wert. Neben den tatsächlich existierenden identifizierten Einflüssen modelliert er also auch sehr viele Einflüsse, die

im zugrundeliegenden Netzwerk nicht existieren. Im Gegensatz dazu begeht der Algorithmus *Strukturlernen in diskreten DBN* nur wenige derartige Fehler und erzielt einen hohen positiv prädiktiven Wert. Allerdings reicht selbst bei großem Datenumfang die darin zur Verfügung gestellte Information für den Algorithmus nicht aus, und er kann nur sehr wenige regulatorische Einflüsse des zugrundeliegenden Netzwerks identifizieren. Wieder sind die Ergebnisse dieser beiden Algorithmen aber insgesamt besser als die des Algorithmus *REM* bei der Arbeit mit einem linearen Additiven Regulationsmodell.

Umfang der gegebenen Daten – Zustandsübergangsdaten

Eine mit Hilfe von Expressionsexperimenten generierte Zeitreihe beschreibt ein spezielles dynamisches Verhalten des zugrundeliegenden Netzwerks und liefert damit prinzipiell weniger Information über die Dynamik des Netzwerks als unabhängige Beobachtungen von Zustandsübergängen, die verschiedene dynamische Prozesse einbeziehen. Ein weiteres Problem bei der Arbeit mit Zeitreihen verdeutlichte das vorangegangene Experiment: Die meisten zur Generierung der Simulationsdaten erzeugten zufälligen Netzwerke gelangten, ausgehend von einem beliebig gewählten Startzustand, zu einem bestimmten Zeitpunkt in einen Attraktor. Es konnten dann keine neuen Netzwerkzustände beobachtet werden. Die jeweiligen Zeitreihen protokollieren ab diesem Zeitpunkt wiederholt das dynamische Verhalten der Netzwerke im Attraktor und können keine zusätzlichen Informationen für die Rekonstruktion der Netzwerke mehr liefern.
Dieser Sachverhalt stellt auch in der Praxis ein Problem dar. Oftmals befindet sich das Genregulationsnetzwerk der zu untersuchenden Zellen bereits in einem Attraktor, und es kann mit Hilfe von Expressionsexperimenten nur das dynamische Verhalten des Netzwerks in diesem speziellen Attraktor charakterisiert werden. Die dabei beobachteten Zustandsübergänge liefern zu wenig Information für die Rekonstruktion eines Genregulationsnetzwerks. Abhilfe schaffen kann hier die transiente Störung des Systems durch eine Variation von extra- und intrazellulären Einflußfaktoren sowie durch transiente Manipulationen bestimmter Gene, die das Genregulationsnetzwerk in einen neuen Zustand führen kann [49]. Wichtig dabei ist, daß die regulatorischen Zusammenhänge des Netzwerks nicht verändert werden. Nach einer Störung gelangt das Netzwerk in einem bestimmten Zeitraum entweder in den gleichen Attraktor, der bereits vor der Störung vorlag, oder in einen neuen Attraktor. Tritt letzteres ein, können Informationen über einen neuen dynamischen Prozeß des Netzwerks gewonnen werden. Mit Hilfe mehrerer derartiger Störungen und dem anschließenden Protokollieren des zeitlichen Expressionsverhaltens können unterschiedliche, kurze Zeitreihen generiert werden. In Kombination miteinander liefern diese eine große Menge an unterschiedlichen Zustandsübergängen des Netzwerks und stellen so mehr Information für die Rekonstruktion des Netzwerks zur Verfügung [13].

Dieses Experiment soll einen Eindruck vermitteln, inwieweit die Ergebnisse des vorangegangenen Experiments verbessert werden können, wenn man den Informationsgehalt der generierten Daten durch die Durchführung von Störungen gezielt steigert. Es wurde dazu idealisiert mit Zustandsübergangsdaten gearbeitet, die jeweils 6, 11, 16, 21, 35, 50, 75 bzw. 100 unterschiedliche, unabhängige Zustandsübergänge charakterisieren. Sie liefern somit die maximale Information über das dynamische Verhalten des zugrundeliegenden Systems, die eine Menge von Zustandsübergängen des jeweiligen Umfangs M überhaupt enthalten kann. Die den Simulationsdaten zugrundeliegenden Netzwerke waren die gleichen, die auch schon zur Generierung der Zeitreihen im vorangegangenen Experiment dienten.

Die erzielten Ergebnisse der Algorithmen sind in Abbildung 5.5 dargestellt. Nicht betrachtet wurde hier der *BPTT* Algorithmus – als Eingabe erwartet dieser Algorithmus explizit eine Zeitreihe.
Die mit dem Additiven Regulationsmodell bzw. mit dem kontinuierlichen DBN arbeitenden Algorithmen zeigen grundsätzlich dasselbe Verhalten wie auch bei der Arbeit mit einer Zeitreihe: Bei zunehmendem Datenumfang steigt sowohl die Sensitivität als auch der positiv prädiktive Wert logarithmisch an. Allerdings fallen die jeweiligen Werte für Sensitivität und positiv prädiktiven Wert hier höher aus. Dies kann mit Hilfe der Tabelle 5.2 nachvollzogen werden, in der die Ergebnisse der Algorithmen in beiden Experimenten vergleichend zusammenfaßt sind. Schon bei einem kleinen Datenumfang von bis zu 21 Zustandsübergängen, bei dem auch die Zeitreihe unterschiedliche Zustandsübergänge garantiert, ist dieser Trend zu erkennen. Bei zunehmendem Datenumfang zeigt sich die Überlegenheit der Ergebnisse aus der Arbeit mit Zustandsübergangsdaten immer deutlicher. Auch bei großem Datenumfang stehen den Algorithmen jetzt neue Informationen über das dynamische Verhalten eines Netzwerks zur Verfügung, und sie können ihre Ergebnisse weiter verbessern. Dies war bei der Arbeit mit einer Zeitreihe nicht möglich, da diese den Algorithmen ab einem bestimmten Zeitpunkt, zu dem das betreffende Netzwerk in einem Attraktor angelangt war, keine neuen Informationen liefern konnte.
Noch besser zeigt sich der schon im vorangegangenen Experiment beobachtete Sachverhalt, daß der *evolutionäre Algorithmus* vor allem bezüglich des positiv prädiktiven Wertes deutlich mehr vom Datenumfang abhängig ist als die anderen Algorithmen dieser Gruppe.

Für den Algorithmus *Reveal* kann analog zur Arbeit mit Zeitreihen auch in diesem Experiment bei steigendem Datenumfang eine Zunahme der Sensitivität verzeichnet werden. Die entsprechenden Werte fallen hier aber etwas geringer aus (vergleiche in Tabelle 5.2). Bei der Diskretisierung der Simulationsdaten können Fehler und Inkonsistenzen entstehen, die es dem Algorithmus erschweren, regulatorische Zusammenhänge zwischen den Netzwerkkomponenten zu identifizieren. Die gegebenen Zustandsübergangsdaten enthalten eine größere Anzahl unterschiedlicher Systemzustände als die gegebene Zeitreihe, die aus vielen gleichen oder zumindest

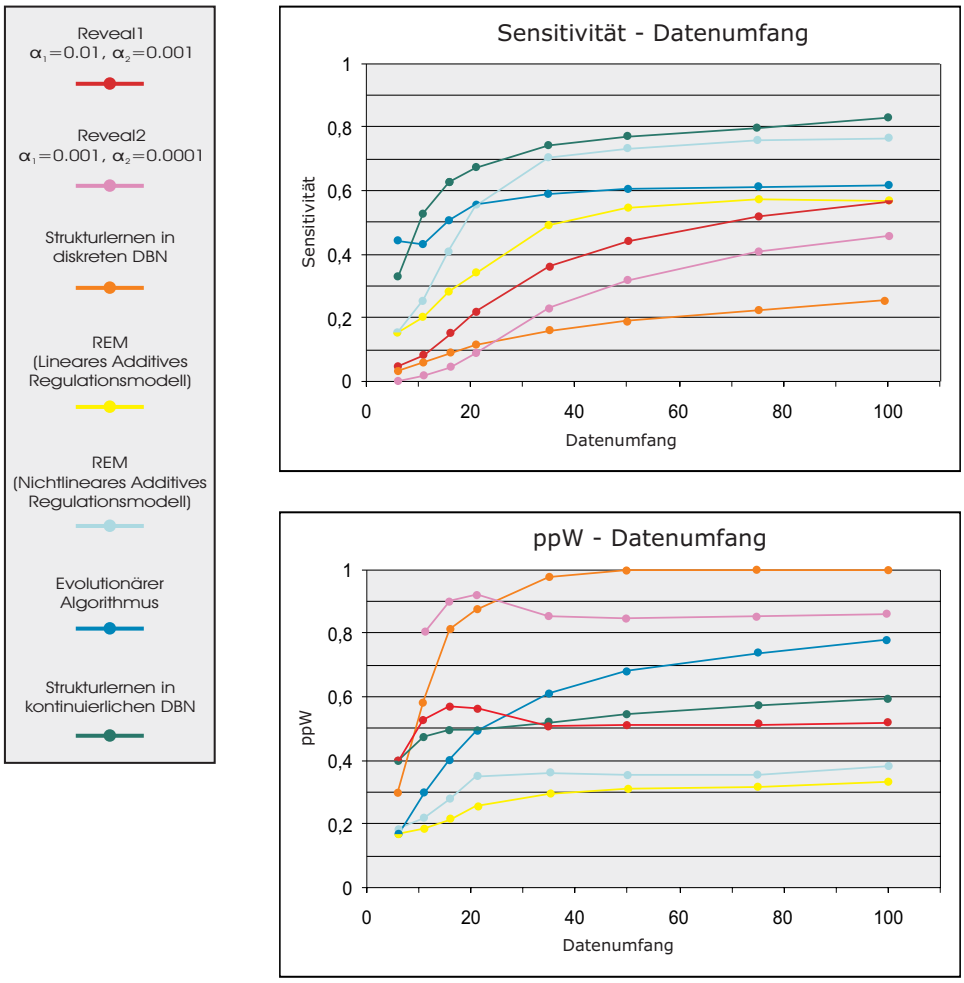


Abbildung 5.5: **Abhängigkeit vom Umfang der gegebenen Daten – Zustandsübergangsdaten.** Jeder Datenpunkt in den Diagrammen entspricht den gemittelten Werten für Sensitivität bzw. positiv prädiktiven Wert über jeweils alle 100 betrachteten Netzwerke. Alle Netzwerke bestanden aus 20 Netzwerkkomponenten; ihr Konnektivität betrug konstant den Wert 3.

ähnlichen kontinuierlichen Netzwerkzuständen besteht. Deshalb können bei der Diskretisierung der Zustandsübergangsdaten im Vergleich zu der Zeitreihe auch mehr Fehler und Inkonsistenzen entstehen. Die Identifizierung von Zusammenhängen ist dann schwerer.

Im Vergleich zum vorangegangenen Experiment kann *Reveal* hier aber einen wesentlich höheren positiv prädiktiven Wert liefern und damit deutlich bessere Ergebnisse erzielen. Gerade die Implementierung *Reveal2*, welche mit kleineren Signifikanzniveaus arbeitet als *Reveal1*, kann auch bei großem Datenumfang einen recht hohen positiv prädiktiven Wert vorweisen. Allerdings bleibt eine leichte Abnahme des po-

	6		11		16		21		35		50		75		100	
	Sens	ppW	Sens	ppW	Sens	ppW	Sens	ppW	Sens	ppW	Sens	ppW	Sens	ppW	Sens	ppW
Reveal1	0.07	0.50	0.16	0.50	0.23	0.47	0.29	0.42	0.39	0.34	0.48	0.26	0.56	0.21	0.60	0.20
	0.04	0.41	0.08	0.53	0.15	0.57	0.22	0.55	0.37	0.50	0.44	0.51	0.52	0.52	0.57	0.52
Reveal2	0.00	-	0.04	0.80	0.11	0.69	0.16	0.62	0.28	0.45	0.36	0.34	0.48	0.26	0.53	0.23
	0.00	-	0.02	0.80	0.04	0.81	0.09	0.92	0.23	0.85	0.31	0.83	0.41	0.87	0.47	0.89
Strukturlernen in diskreten DBN	0.08	0.48	0.11	0.67	0.14	0.71	0.17	0.76	0.19	0.78	0.2	0.77	0.21	0.76	0.22	0.74
	0.03	0.30	0.06	0.6	0.09	0.82	0.11	0.87	0.16	0.98	0.19	0.99	0.22	1.00	0.26	1.00
REM - lineares Additives Regulationsmodell	0.13	0.17	0.19	0.18	0.25	0.19	0.27	0.23	0.35	0.25	0.36	0.25	0.36	0.25	0.38	0.25
	0.16	0.17	0.20	0.19	0.28	0.21	0.34	0.26	0.49	0.29	0.55	0.31	0.57	0.31	0.57	0.33
REM - nichtlineares Additives Regulationsmodell	0.14	0.18	0.22	0.20	0.30	0.24	0.37	0.28	0.47	0.32	0.48	0.32	0.49	0.31	0.49	0.31
	0.16	0.18	0.26	0.22	0.41	0.28	0.55	0.35	0.70	0.36	0.73	0.35	0.76	0.36	0.76	0.38
Evolutionärer Algorithmus	0.46	0.17	0.45	0.28	0.5	0.35	0.51	0.39	0.53	0.43	0.52	0.44	0.51	0.44	0.50	0.46
	0.47	0.16	0.43	0.29	0.51	0.40	0.55	0.49	0.59	0.61	0.61	0.68	0.61	0.74	0.61	0.78
Strukturlernen in kontinuierlichen DBN	0.34	0.39	0.52	0.47	0.59	0.5	0.61	0.50	0.64	0.51	0.64	0.50	0.65	0.49	0.65	0.48
	0.32	0.4	0.53	0.47	0.63	0.49	0.67	0.50	0.74	0.52	0.77	0.55	0.80	0.57	0.81	0.59

Tabelle 5.2: **Tabellarischer Vergleich der Ergebnisse aus den Experimenten zur Untersuchung der Abhängigkeit vom Datenumfang.** Die obere Zeile liefert die Ergebnisse des betreffenden Algorithmus aus der Arbeit mit Zeitreihen. Vergleichend dazu listet die untere Zeile die Ergebnisse aus der Arbeit mit Zustandsübergangsdaten auf.

sitiv prädiktiven Wertes bei zunehmendem Datenumfang bestehen – diese resultiert aus der bereits im vorangegangenen Experiment beschriebenen Zunahme der multiplen Irrtumswahrscheinlichkeit $\alpha_{2,gesamt}$. Ein noch kleineres Signifikanzniveau α_2 könnte dieses Abfallen des positiv prädiktiven Wertes verhindern.

Das Verhalten des Algorithmus *Strukturlernen in diskreten DBN* ist seinem Verhalten im vorangegangenen Experiment ähnlich. Die in den Zustandsübergangsdaten enthaltenen zusätzlichen Informationen bringen hier für eine Verbesserung der Sensitivität nur recht kleine Erfolge. Für einen kleinen Datenumfang müssen sogar geringfügig schlechtere Ergebnisse beobachtet werden (siehe Tabelle 5.2). Verantwortlich dafür sind vermutlich die bereits für den Algorithmus *Reveal* geschilderten Gründe.
Dafür können die in den Zustandsübergangsdaten enthaltenen zusätzlichen Informationen aber den bereits bei der Arbeit mit Zeitreihen recht hohen positiv prädiktiven Wert erheblich steigern. Bei zunehmendem Datenumfang identifiziert der Algorithmus jetzt kaum noch falsche regulatorische Einflüsse.

Zusammenfassend zeigt dieses Simulationsexperiment deutlich die Vorteile der Arbeit mit unabhängigen Zustandsübergangsdaten gegenüber der Verwendung von

Zeitreihen. Natürlich sind unabhängige Zustandsübergangsdaten für praktische Anwendungen eher unrealistisch. Man sollte jedoch versuchen, durch eine geeignete Variation von extra- und intrazellulären Einflußfaktoren oder durch transiente Manipulationen bestimmter Gene unterschiedliche dynamische Prozesse des Genregulationsnetzwerks zu erfassen und damit möglichst viele Informationen über das dynamische Verhalten des Systems zu generieren.
Die Abhängigkeit vom Datenumfang M wird im Vergleich zum vorangegangenen Experiment noch deutlicher. Ein sehr großer Datenumfang ist notwendig, um möglichst viele regulatorische Interaktionen in dem betrachteten Genregulationsnetzwerk zu identifizieren und dabei möglichst wenige Fehler zu begehen.

Meßfehler in den Daten

Expressionsdaten aus biologischen Experimenten unterliegen immer gewissen Meßfehlern. In den bisherigen Experimenten waren die Simulationsdaten deshalb wie in Abschnitt 5.1.2 beschrieben, mit einem Fehler versehen. Mit diesem Simulationsexperiment sollte nun untersuchen werden, wie sich fehlerbehaftete Daten auf die von den verschiedenen Reverse Engineering Methoden erzielten Ergebnisse auswirken.
Zur Generierung der Simulationsdaten wurde mit denselben Netzwerken gearbeitet wie in den vorangegangenen beiden Simulationsexperimenten – sie bestanden aus jeweils 20 Netzwerkkomponenten und besaßen eine konstante Konnektivität von 3. Eine Ausnahme bildet hier wieder die *Adjazenzlisten-Konstruktion*. Da dieser Algorithmus bei einer Konnektivität größer 1 aufgrund von auftretenden Zyklen kaum noch regulatorische Zusammenhänge identifizieren kann und der Einfluß von Meßfehlern damit nicht nachweisbar ist, wurde für diesen Algorithmus auf Netzwerke mit einer Konnektivität von 1 zurückgegriffen.
Die den Reverse Engineering Algorithmen übergebenen Zeitreihen charakterisierten das dynamische Verhalten des jeweils zugrundeliegenden Netzwerks wieder durch 21 verschiedene Zustandsübergänge. Dem Algorithmus *Adjazenzlisten-Konstruktion* mußten stabile Zustandsdaten übergeben werden.
Die Rekonstruktion der Netzwerke erfolgte zunächst mit fehlerfreien Simulationsdaten. Bei einer anschließenden Wiederholung wurden die Simulationsdaten mit einem normalverteilten Fehler versehen. Die für die Erzeugung des Fehlers verwendete Fehlerrate betrug anfänglich 5% und erhöhte sich dann schrittweise auf 10%, 20% bzw. 30%. Analog stieg der maximale Fehler $error_{i,max}$, der einer Expressionsrate $x_i(t)$ zugefügt werden durfte, von 5% auf 10%, 20% bzw. 30% des wahren Wertes $x_i(t)$.

Die graphische Darstellung der Ergebnisse ist in Abbildung 5.6 zu finden. Es ergeben sich folgende Beobachtungen:
Der den Daten zugefügte Fehler hat vor allem auf die Ergebnisse der Algorithmen *REM* (nichtlinearer Ansatz), *BPTT* und *Strukturlernen in kontinuierlichen DBN* sowie auf den *evolutionären Algorithmus* erkennbare Auswirkungen. Da diese Algorithmen mit dem nichtlinearen Additiven Regulationsmodell bzw. dem an diesem

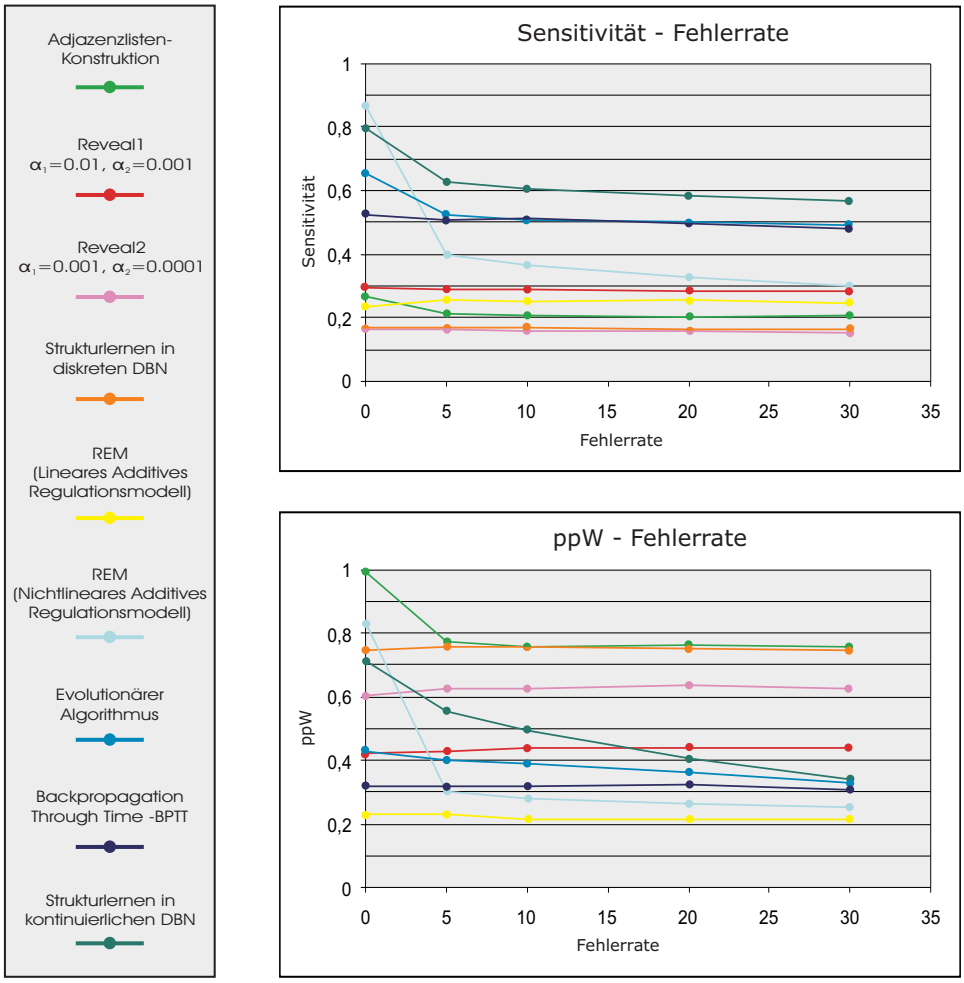


Abbildung 5.6: **Abhängigkeit von Meßfehlern.** Jeder Datenpunkt in den Diagrammen entspricht wieder den über jeweils alle 100 betrachteten Netzwerke gemittelten Werten für Sensitivität bzw. positiv prädiktiven Wert. Alle betrachteten Netzwerke hatten eine konstante Netzwerkgröße N von 20 und eine konstante Konnektivität k von 3. Eine Ausnahme bildet die *Adjazenzlisten-Konstruktion*, deren betrachteten Netzwerke eine Konnektivität von 1 aufwiesen. Den Algorithmen wurde jeweils eine Zeitreihe aus 21 Zustandsübergängen bzw. stabile Zustandsdaten übergeben.

Modell orientierten kontinuierlichen DBN arbeiten, auf dessen Basis auch die Simulationsdaten erzeugt wurden, beeinträchtigen fehlerbehaftete Daten hier die Rekonstruktion des Netzwerks zum Teil erheblich.
Die stärkste Abhängigkeit zeigt der Algorithmus *REM* bei der Arbeit mit einem nichtlinearen Additiven Regulationsmodell. Neben den Meßfehlern wirken hier lediglich Probleme bei der empirischen Bestimmung der maximalen Expressionsraten[8] erschwerend auf die Rekonstruktion der zugrundeliegenden Netzwerke. Der *evolu-*

[8]Siehe Abschnitt *Limitationen* in 3.4.1.

tionäre Algorithmus, der *BPTT* Algorithmus und auch der Algorithmus *Strukturlernen in kontinuierlichen DBN* können bei der Suche nach einem guten Schätzer für die Gewichtsmatrix W bzw. nach einem guten Schätzer der wahren Struktur G in einem lokalen Optimum stecken bleiben. Sowohl für den *evolutionären Algorithmus* als auch für den *BPTT* müssen außerdem algorithmusspezifische Parameter geeignet festgelegt werden. Diese Probleme wirken sich neben Meßfehlern und falsch bestimmten maximalen Expressionsraten zusätzlich negativ auf die Rekonstruktion der Netzwerke aus. Der Einfluß von Meßfehlern fällt hier deshalb geringer aus als bei dem *REM* Algorithmus. Vor allem für den *BPTT* ist nur eine minimale Fehlerabhängigkeit zu erkennen.

Auch für die *Adjazenzlisten-Konstruktion* kann eine Abhängigkeit von Meßfehlern nachgewiesen werden. Hier erschweren fehlerhafte Daten die Konstruktion einer Erreichbarkeitsliste. Die Wahrscheinlichkeit, bei der Durchführung des dafür verwendeten statistischen Tests einen Fehler zu begehen, steigt mit zunehmend größeren Meßfehlern. Falsche Einträge in der Erreichbarkeitsliste führen zu falschen Einträgen in der daraus konstruierten Adjazenzliste; die Sensitivität und der positiv prädiktive Wert sinken.

Für die Algorithmen *Reveal*, *Strukturlernen in diskreten DBN* und *REM* (linearer Ansatz) ist ein Einfluß von Meßfehlern dagegen kaum nachweisbar. Diese Algorithmen arbeiten bei der Modellierung von regulatorischen Einflüssen mit vereinfachenden und falschen Modellannahmen, was die Identifizierung solcher Einflüsse erschwert und die Ergebnisse dieser Algorithmen erheblich beeinträchtigt. Im Vergleich dazu fällt der negative Einfluß von Meßfehlern sehr gering aus und ist hier deshalb praktisch nicht erkennbar.

Ist ein Einfluß der Meßfehler auf die Güte der Ergebnisse für einen Algorithmus nachweisbar, dann wird er vor allem deutlich, nachdem den Simulationsdaten ein Fehler mit einer Fehlerrate von 5% zugefügt wurde. Ein weiteres Ansteigen der Fehlerrate hat dagegen oftmals nur noch relativ geringe Auswirkungen.

Aus diesem Simulationsexperiment läßt sich folgende Schlußfolgerung ziehen: Vereinfachende und falsche Modellannahmen erschweren die korrekte Identifizierung von regulatorischen Einflüssen erheblich. Es konnte gezeigt werden, daß der negative Einfluß von Meßfehlern dagegen sehr gering ausfällt.
Alle in dieser Arbeit betrachteten Reverse Engineering Methoden abstrahieren bei der Modellierung von regulatorischen Einflüssen von der biologischen Realität. Vor allem betrachten sie nur die mRNA-Konzentrationen der Gene und modellieren vereinfachend den Einfluß verschiedener, aus der Expression bestimmter Gene hervorgegangener Proteine auf die Transkription eine Gens durch einen Einfluß der entsprechenden mRNA-Konzentrationen dieser Gene. Für die mit dem Additiven Regulationsmodell arbeitenden Reverse Engineering Algorithmen ergeben sich weiterhin falsche Modellannahmen aus der Unterstellung, daß die einzelnen Einflüsse

verschiedener Gene auf die Expression eines Gens alle additiv und unabhängig wirken. Die Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* arbeiten vereinfachend mit diskreten Expressionsstärken. Diese vereinfachten und auch falschen Betrachtungsweisen der Algorithmen werden bei der Arbeit mit realen Expressionsdaten die Identifizierung von regulatorischen Einflüssen erheblich erschweren, so daß der negative Einfluß von Meßfehlern dagegen wahrscheinlich relativ gering ausfallen wird.
Eine Ausnahme bildet die *Adjazenzlisten-Konstruktion*. Das ihr zugrundeliegende Netzwerkmodell – ein gerichteter Graph – betrachtet lediglich die Struktur des Netzwerks und trifft über dessen Dynamik keine Annahmen. Die Ergebnisse dieses Algorithmus hängen wesentlich davon ab, wie korrekt die Erreichbarkeitsliste konstruiert werden konnte. Wie beschrieben führen größere Meßfehler zu häufigeren Fehlentscheidungen bei der Konstruktion der Erreichbarkeitsliste und zu einer größeren Anzahl an falschen oder fehlenden Einträgen. Der negative Einfluß von Meßfehlern wird also auch bei der Arbeit mit realen Expressionsdaten Auswirkungen auf die von dieser Methode erzielten Ergebnisse zeigen.

Standardabweichungen der Ergebnisse

Die bisher beschriebenen Simulationsexperimente haben untersucht, welche Ergebnisse die verschiedenen Reverse Engineering Algorithmen in Abhängigkeit von wichtigen Eigenschaften der zugrundeliegenden Netzwerke und der verfügbaren Daten liefern. Die Ergebnisse wurden graphisch in Diagrammen veranschaulicht. Jeder einzelne Datenpunkt in diesen Diagrammen beschreibt dabei einen über jeweils alle 100 bzw. 50 betrachteten Netzwerke gemittelten Wert der Sensitivität bzw. des positiv prädiktiven Wertes und dient zur Bewertung des betreffenden Algorithmus. Ein kurzer Blick auf die Standardabweichungen der Ergebnisse soll die Analyse vervollständigen.

Für diese Betrachtung wird nochmals das Simulationsexperiment zur Untersuchung der Abhängigkeit von der Konnektivität der zugrundeliegenden Netzwerke herangezogen. In Abbildung 5.7 sind diesmal die entsprechenden Standardabweichungen der Sensitivität und des positiv prädiktiven Wertes für die verschiedenen Algorithmen in Abhängigkeit von der Konnektivität graphisch dargestellt. Folgende Schlüsse können daraus gezogen werden:
Prinzipiell führt ein höherer Wert für Sensitivität bzw. positiv prädiktiven Wert auch zu einer größeren Standardabweichung. Mit einer zunehmenden Konnektivität konnte ein exponentielles Abfallen der Sensitivität und ein Anstieg des positiv prädiktiven Wertes verzeichnet werden. Ähnliche Beobachtungen ergeben sich hier bezüglich der betreffenden Standardabweichungen. Dies läßt folgenden Schluß zu: Bei einer kleinen Konnektivität setzt sich die gemittelte Sensitivität der meisten Algorithmen aus höheren Sensitivitäten für Netzwerke, in denen recht viele regulatorische Einflüsse identifiziert werden konnten, und aus kleineren Sensitivitäten für Netzwerke, bei

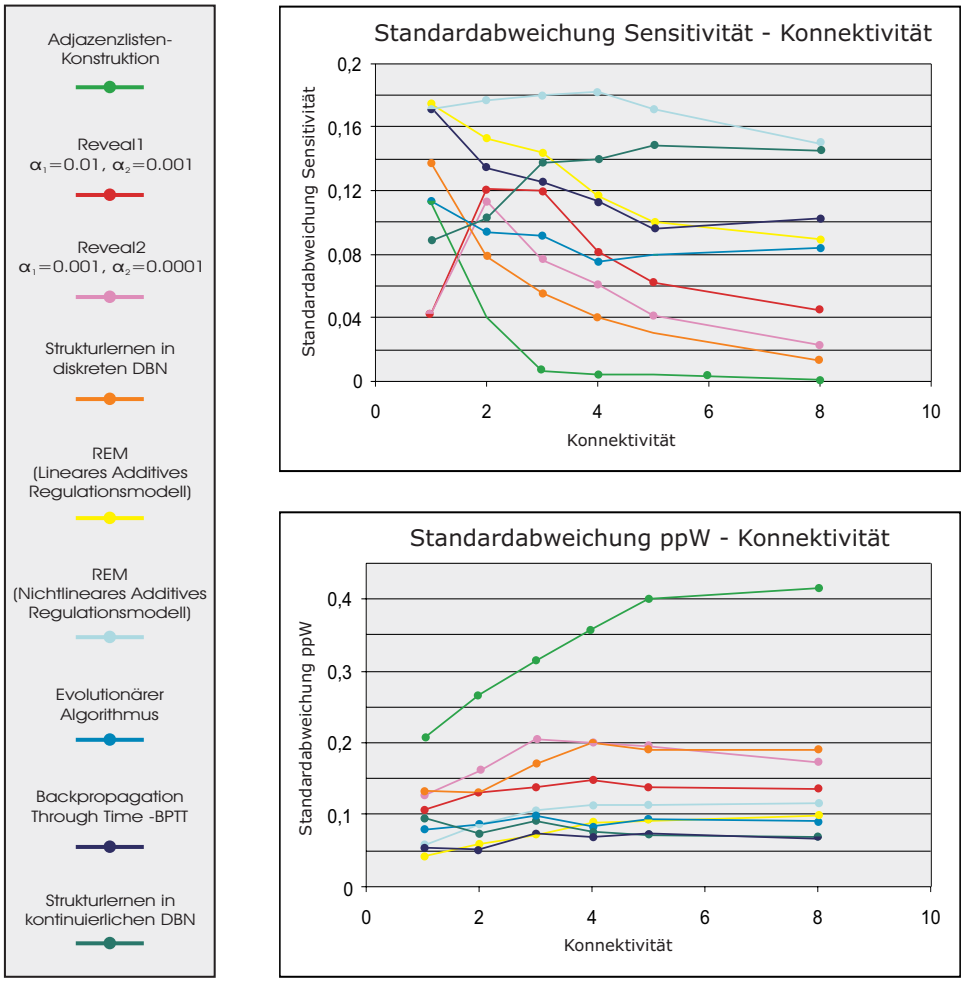


Abbildung 5.7: **Betrachtung der Standardabweichungen.** Für das Experiment zur Untersuchung der Abhängigkeit von der Konnektivität sind hier die entsprechenden Standardabweichungen der erzielten Ergebnisse dargestellt.

denen nur sehr wenige regulatorische Einflüsse nachgewiesen wurden, zusammen. Die Standardabweichung ist groß. Mit zunehmender Konnektivität gibt es immer weniger Netzwerke, für die viele regulatorische Einflüsse aufgedeckt werden können. Die gemittelte Sensitivität sinkt; die entsprechende Standardabweichung ebenfalls. Ein analoges Verhalten ergibt sich bezüglich des positiv prädiktiven Wertes.

Insbesondere fällt auf, daß die Standardabweichung der Sensitivität des Algorithmus *Strukturlernen in kontinuierlichen DBN* – im Gegensatz zu allen anderen Algorithmen – mit zunehmender Konnektivität ebenfalls steigt, obwohl auch für diesen Algorithmus in dem vorangegangenen Experiment ein Abfallen der Sensitivität bei größerer Konnektivität zu beobachten war. Im Gegensatz zu allen anderen Algorith-

men kann der Algorithmus *Strukturlernen in kontinuierlichen DBN* vermutlich bei einer kleinen Konnektivität für fast alle Netzwerke eine konstant hohe Sensitivität liefern. Der entsprechend gemittelte Wert ist dann ebenso relativ hoch; die Standardabweichung recht klein. Bei steigender Konnektivität gelingt es dem Algorithmus für bestimmte Netzwerke zwar nach wie vor, viele regulatorische Zusammenhänge zu finden und eine höhere Sensitivität zu liefern. Mit zunehmender Häufigkeit existieren aber auch Netzwerke, für die der Algorithmus nur wenige Einflüsse nachweisen kann. Die Sensitivität in diesen Netzwerken ist dann recht klein. Die gemittelte Sensitivität sinkt; die Standardabweichung steigt.

Auffällig ist außerdem eine niedrige Standardabweichung der Sensitivität, die der Algorithmus *Reveal* bei einer Konnektivität von 1 liefert. Sie steigt für eine Konnektivität von 2 sprunghaft an und sinkt anschließend wieder ab. Diese Beobachtungen unterstreichen die Ergebnisse des vorangegangenen Experimentes: Bei einer Konnektivität von 1 gelang es dem Algorithmus, in jedem betrachteten Netzwerk nahezu alle existierenden regulatorischen Einflüsse aufzudecken – die gemittelte Sensitivität betrug 99%. Additiv zusammenwirkende Einflüsse ab einer Konnektivität von 2 waren dagegen immer schwerer nachweisbar.

Für die *Adjazenzlisten-Konstruktion* muß – im Vergleich zu allen anderen Algorithmen – eine recht hohe Standardabweichung des positiv prädiktiven Wertes beobachtet werden. Wie in dem Experiment zur Abhängigkeit von der Konnektivität der zugrundeliegenden Netzwerke erläutert, existieren bei zunehmender Konnektivität kaum noch Netzwerkkomponenten, die in keinem Zyklus involviert sind. Die Anzahl nachweisbarer regulatorischer Einflüsse sinkt drastisch. Der Algorithmus kann kaum noch Zusammenhänge identifizieren und die Auswirkung eines falsch identifizierten Einflusses auf den positiv prädiktiven Wert ist sehr groß. Sind die wenigen, für ein Netzwerk identifizierten Einflüsse korrekt, erlangt der Algorithmus in dem entsprechenden Netzwerk einen hohen positiv prädiktiven Wert. Wurde dagegen ein falscher regulatorischer Einfluß identifiziert, ist der positiv prädiktive Wert sehr klein. Dies erklärt die Beobachtung einer entsprechend hohen Standardabweichung in diesem Experiment.

5.3 Kombination der Ergebnisse zweier Algorithmen

Dieser dritte Abschnitt zur Arbeit mit Simulationsdaten soll untersuchen, ob es durch die Kombination der Ergebnisse zweier Algorithmen möglich ist, diese zu verbessern. Es wurde dafür auf die Ergebnisse zurückgegriffen, die die Algorithmen für 100 zufällige Netzwerke der Netzwerkgröße 20 und einer Konnektivität von 3 bei einer jeweils gegebenen Zeitreihe aus 21 Zustandsübergängen liefern. In Tabelle 5.3 findet man einen paarweisen Vergleich der Ergebnisse: Für jeden Algorithmus

		Reveal1 17.4 / 28.2		Strukturlernen diskrete DBN 10.2 / 3.1		REM (linear) 15.9 / 52.7		REM (nichtlinear) 22.2 / 50.5		evolutionärer Algorithmus 31.3 / 50.6		BPTT 30.5 / 66.6	
		truePos	falsePos	truePos	falsePos	truePos	falsePos	truePos	falsePos	truePos	falsePos	truePos	falsePos
Reveal1 17.4 / 28.2	∪	-		-		-		-		-		-	
	∩												
Strukturlernen diskrete DBN 10.2 / 3.1	∪	18.8	29.5	-		-		-		-		-	
	∩	8.8	1.82										
REM (linear) 15.9 / 52.7	∪	18.8	29.5	21.5	55.1	-		-		-		-	
	∩	8.8	6.15	4.6	0.8								
REM (nichtlinear) 22.2 / 50.5	∪	31.6	73.7	27.3	53.0	28.4	84.4	-		-		-	
	∩	8.0	4.9	5.0	0.6	9.7	18.8						
evolutionärer Algorithmus 31.3 / 50.6	∪	36.9	73.4	33.7	53.2	37.6	90.8	37.9	88.9	-		-	
	∩	11.8	5.4	7.7	0.6	9.6	12.5	15.5	12.2				
BPTT 30.5 / 66.6	∪	33.2	82.6	31.3	67.9	36.6	108.6	39.1	107.8	42.1	105.8	-	
	∩	14.8	12.2	9.4	1.2	9.8	10.6	13.6	9.2	19.7	11.4		
Strukturlernen kontinuierliche DBN 36.8 / 38.1	∪	39.5	61.0	37.4	40.5	42.1	83.3	42.0	81.8	42.8	78.2	42.5	93.5
	∩	14.7	5.3	9.6	0.8	10.6	7.5	17.0	6.8	25.3	10.5	24.9	11.2

Tabelle 5.3: **Kombination der Ergebnisse zweier Algorithmen:** Für jeden Algorithmus ist im jeweiligen Spalten- bzw. Zeilenkopf sein eigenes Ergebnis angeben – der erste Wert beschreibt die durchschnittliche Anzahl $truePos$ korrekt identifizierter regulatorischer Einflüsse und der zweite, farblich hervorgehobene Wert die durchschnittliche Anzahl $falsePos$ aller falsch identifizierten regulatorischen Einflüsse. Für je zwei Algorithmen wurde zunächst die Vereinigung ($\cup$) der von beiden Algorithmen identifizierten regulatorischen Einflüsse betrachtet. Der linke obere Wert $truePos$ gibt die durchschnittliche Anzahl der korrekt identifizierten regulatorischen Einflüsse und der rechte obere Wert die durchschnittliche Anzahl falsch identifizierter regulatorischer Einflüsse dieser vereinigten Menge an. Die zweite Zeile betrachtet anschließend analog die Schnittmenge ($\cap$) der von beiden Algorithmen identifizierten regulatorischen Einflüsse.

ist im zugehörigen Spalten- bzw. Zeilenkopf sein eigenes Ergebnis angeben. Der erste Wert bezieht sich dabei auf den Wert $truePos$ und gibt an, wieviele korrekte regulatorische Einflüsse – also Einflüsse, die in dem den jeweiligen Daten zugrundeliegenden Netzwerk tatsächlich existieren – der Algorithmus durchschnittlich in den 100 betrachteten Netzwerken identifizieren konnte. Der zweite, hier rot hervorgehobene Wert $falsePos$ beschreibt dagegen die durchschnittliche Anzahl aller vom Algorithmus falsch identifizierten regulatorischen Einflüsse.
Die Tabelle analysiert für je zwei Algorithmen die Kombination ihrer Ergebnisse. Dazu wird zunächst die Vereinigung der von beiden Algorithmen identifizierten re-

gulatorischen Einflüsse betrachtet. Der linke obere Wert $truePos$ gibt die Anzahl der korrekt identifizierten regulatorischen Einflüsse an, die in dieser vereinigten Menge enthalten sind. Dagegen beschreibt der rechte obere Wert die Anzahl falsch identifizierter regulatorischer Einflüsse dieser vereinigten Menge. Trivialerweise sind die Werte für $truePos$ und $falsePos$ der Vereinigung mindestens so groß wie die entsprechenden Werte der Einzelergebnisse beider Algorithmen. Die zweite Zeile betrachtet anschließend die Schnittmenge der von beiden Algorithmen identifizierten regulatorischen Einflüsse. Die beiden unteren Werte für $truePos$ und $falsePos$ geben analog zu den oberen Werten die Anzahl aller korrekt bzw. falsch identifizierten regulatorischen Einflüsse dieser Schnittmenge an. Im Gegensatz zur Vereinigung können die Werte $truePos$ und $falsePos$ der Schnittmenge nur maximal so groß sein, wie der jeweils entsprechende Wert des diesbezüglich schlechteren Einzelergebnisses.
Die Beobachtung folgender zwei Fälle wäre vorteilhaft:

1. Es kann für zwei Algorithmen festgestellt werden, daß sie recht unterschiedliche korrekt identifizierte regulatorische Einflüsse liefern; bei der Identifizierung von falschen regulatorischen Einflüsse begehen sie dagegen die gleichen Fehler:

   Der Wert $truePos$ für die Schnittmenge der von den Algorithmen identifizierten regulatorischen Einflüsse wäre dann im Vergleich zu den Einzelergebnissen der Algorithmen recht klein; der entsprechende Wert von $falsePos$ dagegen recht groß – idealerweise entspricht dieser annähernd dem kleineren Wert von $falsePos$ der Einzelergebnisse der Algorithmen. Eine Vereinigung der von den beiden Algorithmen identifizierten regulatorischen Einflüsse würde dann in einer höheren Sensitivität resultieren.

2. Es kann für zwei Algorithmen beobachtet werden, daß sie zwar die gleichen korrekt identifizierten regulatorischen Einflüsse liefern, dagegen aber recht unterschiedliche falsch identifizierte regulatorische Einflüsse:

   Die Schnittmenge der von beiden Algorithmen identifizierten regulatorischen Einflüsse würde sich dann durch einen recht hohen Wert von $truePos$ auszeichnen und annähernd dem kleineren Wert von $truePos$ der Einzelergebnisse entsprechen. Der Wert $falsePos$ wäre dagegen im Vergleich zu den Einzelergebnissen der Algorithmen recht klein. Diese Schnittmenge der von den beiden Algorithmen identifizierten regulatorischen Einflüsse würde dann in einem höheren positiv prädiktiven Wert resultieren. Die entsprechende Sensitivität wäre zumindest nicht wesentlich niedriger als die kleinere Sensitivität der beiden Einzelergebnisse.

Die genauere Betrachtung von Tabelle 5.3 läßt einen Trend zum zweiten Fall erkennen. Die Werte $falsePos$ der paarweise gebildeten Schnittmengen (in Tabelle 5.3 farblich unterlegt) sind relativ klein und zum Teil erheblich niedriger als die entsprechenden Werte der Einzelergebnisse (ebenfalls farblich gekennzeichnet). Bezüglich

der Anzahl an korrekt identifizierten regulatorischen Einflüssen müssen bei der Bildung der Schnittmenge gewisse Verschlechterungen akzeptiert werden. Trotzdem ist zu erkennen, daß die Algorithmen viele gleiche korrekte regulatorische Einflüsse nachweisen. Der aus der Schnittmengenbildung resultierende Verlust an korrekt identifizierten regulatorischen Einflüssen fällt geringer aus als die Verringerung der – in den Einzelergebnissen zum Teil recht hohen – Anzahl an falsch identifizierten regulatorischen Einflüssen, so daß zumindest in einigen Fällen die Ergebnisse der Kombination insgesamt besser ausfallen als die jeweiligen Einzelergebnisse. Beispielsweise enthält die entsprechende Kombination der von dem *evolutionären Algorithmus* identifizierten regulatorischen Einflüsse mit denen des Algorithmus *Strukturlernen in kontinuierlichen DBN* durchschnittlich immer noch 25.3 korrekte regulatorische Einflüsse, was einen Verlust von 11.5 bzw. 6 korrekt identifizierten Einflüssen gegenüber den Einzelergebnissen bedeutet. Gleichzeitig kann aber auch die Anzahl falsch identifizierter regulatorischer Einflüsse von 38.1 bzw. 50.6 auf nur 10.5 gesenkt werden.

5.4 Integration von Vorwissen

Abschließend soll dieser letzte Abschnitt zur Arbeit mit Simulationsdaten an einem kleinen Beispiel zeigen, wie sich durch die Integration von Vorwissen über die Struktur des zu rekonstruierenden Netzwerks der Reverse Engineering Prozeß sinnvoll unterstützen läßt.
In Kapitel 4 wurde einleitend erörtert, wie Vorwissen überhaupt erlangt werden kann. Eine Möglichkeit hierbei war es, auf die Ergebnisse aus gezielten Experimenten, auf Angaben aus der Literatur und auf biologisches Expertenwissen zurückzugreifen, um einzelne regulatorische Einflüsse zu charakterisieren. Eine zweite Möglichkeit ergab sich durch die Verwendung genomweiter Manipulations- und Expressionsexperimente, die es erlaubt, eine Vielzahl der prinzipiell möglichen regulatorischen Einflüsse vor dem Reverse Engineering Prozeß auszuschließen. Die Generierung von Vorwissen für dieses Experiment mit Simulationsdaten orientierte sich an der zweiten Variante. Um keine speziellen Annahmen über den genauen praktischen Ablauf der Manipulations- und Expressionsexperimente treffen zu müssen, wurde hier allerdings auf einer abstrakten Ebene gearbeitet. Die wahre Struktur eines Netzwerks bildete dabei die Vorlage zur Erzeugung einer Struktur G_{prior}, die das generierte Vorwissen repräsentierten sollte: Wie für die zweite Variante erläutert, können existierende regulatorische Einflüsse relativ sicher nachgewiesen werden. Ein tatsächlich im zugrundeliegenden Netzwerk existierender regulatorischer Einfluß wurde deshalb mit einer Wahrscheinlichkeit von 80% in die Struktur G_{prior} aufgenommen. Um gleichzeitig den Sachverhalt zu berücksichtigen, daß neben den tatsächlich existierenden Einflüssen auch falsche Zusammenhänge aufgrund von indirekten Beziehungen beobachtet werden (siehe Kapitel 4), wurde ein nicht

existierender regulatorischer Einfluß mit einer Wahrscheinlichkeit von 20% in G_{prior} eingefügt. Als Grundlage für dieses Simulationsexperiment dienten die bereits in den vorangegangenen Experimenten erzeugten 100 Modellnetzwerke mit einer Netzwerkgröße N von 20 und einer Konnektivität k von 3. Die jeweils generierte Struktur G_{prior} enthielt im Durchschnitt 48 der 60 in einem solchen Modellnetzwerk existierenden regulatorischen Einflüsse. Neben diesen korrekten Zusammenhängen beschrieb sie aber auch 67 falsche regulatorische Einflüsse, die in dem jeweils zugrundeliegenden Netzwerk nicht existierten. Wie also für die zweite Variante (siehe Kapitel 4) beschrieben, war die tatsächliche Existenz der in G_{prior} enthaltenen regulatorischen Zusammenhänge nicht sicher – nur 48 der 115 vermuteten Zusammenhänge waren korrekt. Nicht in G_{prior} beschriebene regulatorische Einflüsse konnten dafür mit einer hohen Wahrscheinlichkeit ausgeschlossen werden – dabei beging man nur in 12 von 285[9] Fällen einen Fehler.

Das durch G_{prior} repräsentierte Vorwissen wurde mit Hilfe der in Kapitel 4 beschriebenen Ansätze zur Integration von Vorwissen bei der Arbeit mit Booleschen Netzwerken, Additiven Regulationsmodellen bzw. Dynamischen Bayesschen Netzwerken in einen entsprechenden Reverse Engineering Algorithmus jeweils wie folgt integriert:
Für den Algorithmus *Reveal* galt es, mit Hilfe dieses Vorwissens jedem Paar (x_j, x_i) zweier Netzwerkkomponenten die Wahrscheinlichkeit der Hypothese $H_{1,(x_j,x_i)}$ zuzuordnen, daß x_j einen regulatorischen Einfluß auf x_i ausübt. Ist in G_{prior} kein regulatorischer Einfluß von x_j auf x_i beschrieben, so existiert dieser mit einer hohen Wahrscheinlichkeit tatsächlich nicht. Der Wahrscheinlichkeit $P(H_{1,(x_j,x_i)})$ wurde deshalb auf einen sehr kleinen Wert von 0.05 festgelegt. Schlägt G_{prior} dagegen einen regulatorischen Einfluß von x_j auf x_i vor, so ist dieser zwar wahrscheinlicher als ohne das entsprechende Wissen aus G_{prior}, aber dennoch nicht sicher. Die Wahrscheinlichkeit $P(H_{1,(x_j,x_i)})$ bekam deshalb den Wert 0.7 zugewiesen[10].
Die Integration von Vorwissen bei der Arbeit mit einem nichtlinearen Additiven Regulationsmodell wurde an dem Algorithmus *REM* getestet. Mit Hilfe des Vorwissens wurden hier die entsprechenden Gewichte w_{ij} all der regulatorischen Einflüsse, die nicht in G_{prior} enthalten waren, zu Beginn des Reverse Engineering Prozesses auf den Wert 0 festgelegt und so das Wissen ausgenutzt, daß diese regulatorischen Einflüsse relativ sicher nicht existieren. Vorwissen bezüglich der übrigen Einflüsse wurde nicht betrachtet, denn wie in Abschnitt 4.2 beschrieben, sollte bei diesem Ansatz nur sehr sicheres Vorwissen integriert werden, da sich keine Möglichkeit bietet, qualitative Angaben über die Sicherheit des Vorwissens zu berücksichtigen.
Bei der Arbeit mit Dynamischen Bayesschen Netzwerken entsprach die Struktur G_{prior} der wahrscheinlichsten Struktur G^* bezüglich des Vorwissens. Für jeden mög-

[9] In einem Netzwerk der Größe $N = 20$ und der Konnektivität 3 gibt es insgesamt 400 potentielle regulatorische Einflüsse. G_{prior} schließt 285 dieser Einflüsse aus.

[10] Die Werte 0.05 und 0.7 für die Wahrscheinlichkeiten $P(H_{1,(x_j,x_i)})$ wurden intuitiv festgelegt.

lichen regulatorischen Einfluß mußte eine geeignete Konstante κ_{ij} definiert werden, welche die Sicherheit des in G^* beschriebenen Vorwissens über diesen Einfluß spezifiziert. Dazu wurde auch hier die Tatsache berücksichtigt, daß man einen regulatorischen Einfluß relativ sicher ausschließen kann, wenn er nicht in G^* enthalten ist. Die ihm zugeordnete Konstante κ_{ij} bekam deshalb den Wert 0.05 zugewiesen. Umgekehrt wurde die Konstante κ_{ij} auf den Wert 0.6 festgelegt, wenn G^* den zugehörigen Einfluß enthielt[11]. Das Simulationsexperiment zu diesem Ansatz erfolgte auf der Basis des Algorithmus *Strukturlernen in kontinuierlichen DBN*.

In Abbildung 5.8 sind die Ergebnisse zu finden, die die drei Algorithmen *Reveal*, *REM* und *Strukturlernen in kontinuierlichen DBN* bei der Rekonstruktion der betrachteten Modellnetzwerke mit Hilfe der jeweils übergebenen Zeitreihe eines Datenumfangs von 6, 11, 16 bzw. 21 und unter Berücksichtigung des Vorwissens in G_{prior} durchschnittlich lieferten (durchgezogene Linie). Zum Vergleich sind außerdem die Ergebnisse dargestellt, die die Algorithmen ohne eine Integration des Vorwissens erzielten (durchbrochene Linie). Wie zu erkennen, gelang es den Algorithmen, durch die geeignete Integration des Vorwissens sowohl die Sensitivität als auch den positiv prädiktiven Wert zum Teil recht erheblich zu steigern.
Die größte Verbesserung konnte der Algorithmus *REM* erzielen. Ist kein Vorwissen über die Struktur eines Netzwerks vorhanden, arbeitet er bei der Rekonstruktion mit einem vollständig verknüpften Netzwerk. Sehr viele Gewichte – in einem Netzwerk der Größe N beträgt die Anzahl $N(N+1)$ – müssen dann geschätzt und klassifiziert werden. Deshalb lieferte dieser Algorithmus in den vorangegangenen Experimenten zum Teil recht unbefriedigende Ergebnisse. Mit Hilfe des Vorwissens konnte der Algorithmus hier schon zu Beginn sehr viele Gewichte korrekt in die Klasse „null“ einordnen und mußte diese im Reverse Engineering Prozeß nicht berücksichtigen. Die Anzahl der zu schätzenden Gewichte sinkt erheblich; die Genauigkeit der einzelnen Schätzungen steigt und fördert ein korrektes Klassifizieren. Dies wirkt sich sehr positiv sowohl auf die Sensitivität als auch auf den positiv prädiktiven Wert aus.
Im Gegensatz zum Algorithmus *REM* verbindet der Algorithmus *Strukturlernen in kontinuierlichen DBN* die Schätzung der Gewichte mit einem expliziten Erlernen der Struktur. Durch seinen wahrscheinlichkeitstheoretischen Hintergrund gelingt es ihm außerdem besser, mit Inkonsistenzen und Meßfehlern in den Daten umzugehen. Deshalb konnte er in den vorangegangenen Experimenten zum Teil wesentlich bessere Ergebnisse liefern als der Algorithmus *REM*. Die Steigerungen der Ergebnisse durch die Integration von Vorwissen fallen hier geringer aus als bei *REM*, werden aber dennoch deutlich. Der positiv prädiktive Wert erfährt dabei eine größere Steigerung als die Sensitivität, denn mit Hilfe des bereitgestellten Vorwissens konnte vor allem die Tatsache unterstützt werden, daß viele der potentiellen regulatorischen Einflüsse nicht existieren.
Eine ganz ähnliche Beobachtung ergibt sich für den Algorithmus *Reveal*. Gerade hier

[11] Die Werte 0.05 und 0.6 für die Parameter κ_{ij} wurden intuitiv festgelegt.

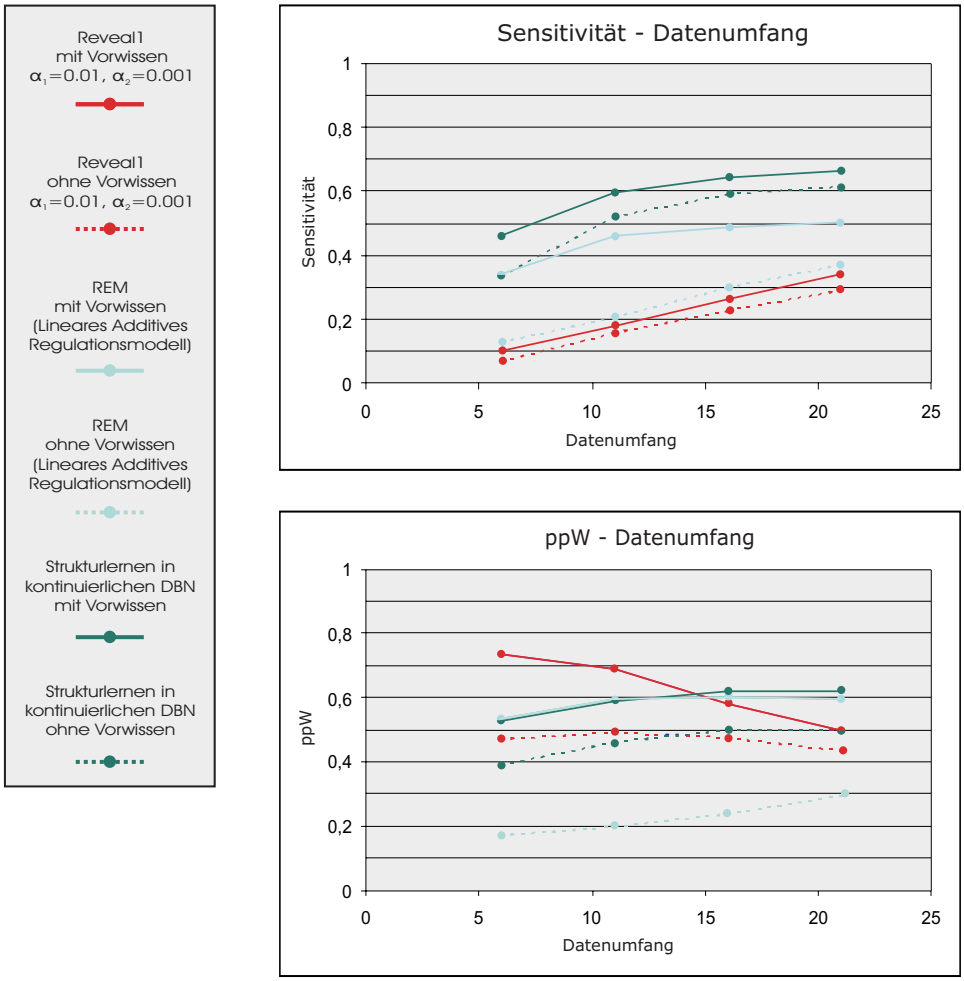


Abbildung 5.8: **Integration von Vorwissen.** Es sind die Ergebnisse dargestellt, die die Algorithmen bei der Rekonstruktion der betrachteten 100 Modellnetzwerke mit Hilfe der jeweils übergebenen Zeitreihe eines Datenumfangs von 6, 11, 16 bzw. 21 und unter Berücksichtigung des Vorwissens durchschnittlich lieferten (durchgezogene Linie). Zum Vergleich sind ebenfalls die Ergebnisse beschrieben, die die Algorithmen ohne eine Integration des Vorwissens lieferten (durchbrochene Linie).

sind bezüglich der Sensitivität seiner Ergebnisse nach der Integration von Vorwissen nur geringe Verbesserungen zu verzeichnen. Aufgrund der Unsicherheit über die tatsächliche Existenz der in G_{prior} vorgeschlagenen regulatorischen Einflüsse wurden diese im Reverse Engineering Prozeß nur bedingt unterstützt. Dafür konnte der *Reveal* Algorithmus genau wie der Algorithmus *Strukturlernen in kontinuierlichen DBN* den positiv prädiktiven Wert seiner Ergebnisse steigern, denn das Wissen über nicht existente regulatorische Zusammenhänge in G_{prior} war recht sicher und wurde im Reverse Engineering Prozeß entsprechend unterstützt. Deutlich wird bei

der Betrachtung des positiv prädiktiven Wertes auch eine bereits in vorangegangenen Experimenten beobachtete, typische Eigenschaft des Algorithmus: Bei steigendem Datenumfang fällt der positiv prädiktive Wert. Dies war auf den Sachverhalt zurückzuführen, daß es mit umfangreicheren Daten generell besser gelingt, regulatorische Zusammenhänge zwischen einer Menge von Inputelementen X und einem Outputelement x_i durch den χ^2-Unabhängigkeitstest nachzuweisen. Dies wiederum bewirkt die häufigere Ausführung eines zweiten χ^2-Unabhängigkeitstests, mit dem für jedes Element aus X überprüft werden muß, ob es tatsächlich benötigt wird, um die Abhängigkeit des Elements x_i von der Menge X festzulegen. Es ergibt sich eine größere multiple Irrtumswahrscheinlichkeit $\alpha_{2,gesamt}$, und der positiv prädiktive Wert sinkt (vergleiche auch Abschnitt 5.2). Da der Nachweis von regulatorischen Einflüssen – wenn auch nur gering – bei der Arbeit mit Vorwissen zusätzlich unterstützt wird, ist dieser Effekt hier verstärkt zu beobachten.

5.5 Zusammenfassung der Ergebnisse

In diesem 5. Kapitel wurden die in Tabelle 5.1 zusammengefaßten Reverse Engineering Methoden an Simulationsdaten getestet und die Ergebnisse ausführlich diskutiert. Die wichtigsten daraus resultierenden Erkenntnisse sollen hier – auch in Hinblick auf die Arbeit mit realen Expressionsdaten – zusammengefaßt werden.

Zunächst wurde in Abschnitt 5.2 der Einfluß wichtiger Netzwerkparameter und bestimmter Eigenschaften der Daten auf die Güte der Ergebnisse der einzelnen Reverse Engineering Methoden untersucht. Im wesentlichen ergaben sich dabei folgende Beobachtungen:
Mit zunehmender Konnektivität lassen sich regulatorische Einflüsse schwerer identifizieren; die Sensitivität der Algorithmen sinkt exponentiell. Erfährt eine Netzwerkkomponente nur einen einzigen regulatorischen Einfluß, kann dieser relativ gut nachgewiesen werden. Steigt die Anzahl der auf eine Netzwerkkomponente wirkenden regulatorischen Einflüsse, läßt sich ein einzelner dieser Einflüsse viel schwerer aufdecken. Es ist zu vermuten, daß dieser Aspekt bei der Arbeit mit realen Expressionsdaten noch deutlicher wird. Bei der Generierung der Simulationsdaten wurde vereinfachend angenommen, daß alle Einflüsse additiv und vor allem unabhängig voneinander auf die betreffende Netzwerkkomponente einwirken. Gerade die Unabhängigkeit ist biologisch jedoch nicht realistisch. Wirken zwei regulatorische Einflüsse nicht unabhängig voneinander, wird das ihre Identifizierung zusätzlich erschweren, denn sie sind dann nur als Kombination nachweisbar.
Bei zunehmender Netzwerkgröße müssen die Algorithmen eine größere Menge an potentiellen regulatorischen Einflüssen betrachten, um die auf eine Netzwerkkomponente einwirkenden Einflüsse festzulegen. Die Wahrscheinlichkeit, einen Fehler zu begehen und einen falschen regulatorischen Einfluß zu identifizieren, steigt. Die Algorithmen liefern dann neben den korrekt identifizierten regulatorischen Ein-

flüssen auch eine zunehmende Anzahl an falsch identifizierten regulatorischen Zusammenhängen. Bei der Untersuchung des Einflusses einer steigenden Netzwerkgröße wurde deshalb ein exponentiell abfallender positiv prädiktiver Wert beobachtet.
Die Simulationsexperimente zur Untersuchung der Abhängigkeit vom gegebenen Datenumfang bestätigten die Erwartungen, daß eine sehr große Anzahl an Zustandsübergängen notwendig sein wird, um bei der Rekonstruktion eines Netzwerks gute Ergebnisse zu erzielen und viele der tatsächlich existierenden Einflüsse korrekt zu identifizieren. Es ist außerdem wichtig, daß die gegebenen Daten möglichst unterschiedliche Zustandsübergänge enthalten, verschiedene zeitliche Prozesse charakterisieren und damit den Algorithmen sehr viele Informationen über das dynamische Verhalten des zugrundeliegenden Systems liefern. Dies wurde besonders im Vergleich der Ergebnisse aus der Arbeit mit Zeitreihen und der Arbeit mit unabhängigen Zustandsübergängen deutlich.
Eine weitere Beobachtung konnte in dem Simulationsexperiment zur Untersuchung des Einflusses fehlerbehafteter Daten gewonnen werden: Der negative Einfluß von Meßfehlern auf die Güte der Ergebnisse fällt im Vergleich zu den Schwierigkeiten, die sich durch vereinfachte und falsche Modellannahmen ergeben, eher gering aus. Es ist deshalb zu erwarten, daß der negative Einfluß von Meßfehlern bei der Arbeit mit realen Expressionsdaten nur bedingt zum Tragen kommt, da die in dieser Arbeit betrachteten Reverse Engineering Methoden bei der Modellierung eines Genregulationsnetzwerks von der biologischen Realität abstrahieren und diese vereinfachten und falschen Modellannahmen den Reverse Engineering Prozeß bei der Arbeit mit realen Expressionsdaten erheblich erschweren werden.

Im weiteren Verlauf dieses Kapitels wurde anschließend die kombinierte Betrachtung der Ergebnisse zweier Algorithmen analysiert. Diese Untersuchungen ergaben, daß durch die Bildung der Schnittmenge der von zwei Algorithmen identifizierten regulatorischen Einflüsse vor allem die Anzahl der falsch identifizierten regulatorischen Interaktionen verringert werden kann. Die tatsächliche Existenz der in der Schnittmenge enthaltenen regulatorischen Einflüsse ist sehr viel wahrscheinlicher, als die Existenz der regulatorischen Zusammenhänge in den Einzelergebnissen beider Algorithmen.

Der letzte Abschnitt 5.4 konnte abschließend zeigen, daß sinnvoll eingesetztes Vorwissen das Reverse Engineering unterstützen und so die Ergebnisse der Algorithmen verbessern kann.

Wie erläutert, ist ein direkter Vergleich der Ergebnisse aller Algorithmen nicht möglich. Die Algorithmen *REM* (nichtlinearer Ansatz), *BPTT* und *Strukturlernen in kontinuierlichen DBN* sowie der *evolutionäre Algorithmus* arbeiten auf Grundlage des nichtlinearen Additiven Regulationsmodell bzw. des an diesem Modell orientierten kontinuierlichen DBN. Da das nichtlineare Additive Regulationsmodell auch die Basis für die Generierung der Simulationsdaten bildete, kamen wichtige modellbe-

dingte Limitationen dieser Algorithmen in den Experimenten nicht zum Tragen, und der Reverse Engineering Prozeß wurde ihnen im wesentlichen lediglich durch Meßfehler und Probleme bei der empirischen Bestimmung der maximalen Expressionsraten erschwert. Für die übrigen Algorithmen ergaben sich außerdem Schwierigkeiten durch vereinfachende und falsche Modellannahmen, die das Reverse Engineering zusätzlich beeinträchtigten. Ein Vergleich der Ergebnisse mußte deshalb getrennt für die einzelnen Gruppen erfolgen:
Der Algorithmus *Strukturlernen in kontinuierlichen DBN* lieferte in allen Experimenten bessere Ergebnisse als die auf dem nichtlinearen Additiven Regulationsmodell basierenden Algorithmen. Seine Überlegenheit resultiert vor allem aus der Tatsache, daß er die Schätzung der Gewichtsmatrix W mit einer expliziten Suche nach der wahren Struktur verbindet und dadurch den Reverse Engineering Prozeß maßgeblich unterstützt. Auch kann er aufgrund der wahrscheinlichkeitstheoretischen Natur des kontinuierlichen DBN besser mit Fehlern und Inkonsistenzen in den Daten umgehen. Die Ergebnisse des *evolutionären Algorithmus* waren im allgemeinen etwas besser als die des *BPTT* Algorithmus; vor allem bezüglich des positiv prädiktiven Wertes konnte er bessere Werte erzielen. Dem Algorithmus *REM* (nichtlinearer Ansatz) gelang es dagegen im allgemeinen nicht, sich gegen die anderen Algorithmen dieser Gruppe durchzusetzen.
Die vereinfachende Annahme diskreter Expressionsraten erschwerte den Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* das Identifizieren von regulatorischen Einflüssen zum Teil erheblich. Der Grund hierfür lag vor allem in dem aus der Diskretisierung der Daten resultierenden Informationsverlust und den entstehenden Inkonsistenzen. Bei steigendem Datenumfang gelang es zumindest dem Algorithmus *Reveal*, seine Sensitivität erheblich zu steigern, während dem Algorithmus *Strukturlernen in diskreten DBN* auch bei großem Datenumfang für den Reverse Engineering Prozeß nicht genügend Informationen zur Verfügung standen. Dafür konnte dieser Algorithmus oftmals einen sehr hohen positiv prädiktiven Wert vorweisen. Bei der Arbeit mit einem linearen Additiven Regulationsmodells erzielte der Algorithmus *REM* häufig nur einen recht kleinen positiv prädiktiven Wert, und seine Ergebnisse fielen deshalb im allgemeinen schlechter aus, als die der anderen beiden Algorithmen.
Die Ergebnisse der *Adjazenzlisten-Konstruktion* waren in den vorangegangenen Experimenten zum Teil recht unbefriedigend. Aufgrund von Zyklen in den zugrundeliegenden Netzwerken gelang es ihr bereits ab einer Konnektivität von 1 kaum noch, regulatorische Einflüsse zu identifizieren; die Sensitivität erreichte schnell den Wert 0. Wie bereits beschrieben, muß hier berücksichtigt werden, daß die in den Simulationsexperimenten betrachteten Konnektivitäten zwar durchaus realistisch für ein reales Genregulationsnetzwerk sind, dafür aber eine Netzwerkgröße von 20 Netzwerkkomponenten viel zu klein gewählt ist. Reale Genregulationsnetzwerke sind in der Regel sehr viel größer. In größeren, zufällig konstruierten Netzwerken sinkt bei gleichbleibender Konnektivität die Wahrscheinlichkeit, das eine Netzwerkkomponente in einem Zyklus involviert ist. Damit steigt die Anzahl nachweisbarer regulato-

rischer Einflüsse und der Algorithmus kann eine höhere Sensitivität liefern. Dieser Sachverhalt läßt hoffen, daß der Algorithmus bei der Arbeit mit realen Expressionsdaten bessere Ergebnisse erzielen kann. Allerdings besitzen Genregulationsnetzwerke vielleicht andere strukturelle Eigenschaften als zufällig konstruierte Netzwerke und bevorzugen gerade die Anordnung der Netzwerkkomponenten in Zyklen, wodurch sich die Anzahl nachweisbarer regulatorischer Einflüsse zusätzlich verringert.

Die vorangegangenen Simulationsexperimente erlauben zwar Aussagen über das grundlegende Verhalten der Algorithmen bei der Arbeit mit realen Expressionsdaten; die dabei im einzelnen erzielten Ergebnisse lassen sich aber nicht direkt übertragen. Da die Simulationsdaten auf dem nichtlinearen Additiven Regulationsmodell basieren, kamen jeweils wesentliche modell- und algorithmusbedingte Limitationen der einzelnen Algorithmen in den vorangegangen Experimenten nicht zum Tragen:
Die wohl entscheidendste Limitation der Algorithmen ergibt sich aufgrund der vereinfachten Annahme, daß Genregulationsprozesse nur auf der Ebene der Transkription stattfinden. Alle Regulationsprozesse auf anderen Ebenen der Genexpression werden ignoriert und eine starke Korrelation zwischen Protein- und mRNA-Konzentration eines Gens angenommen. Um die Expressionsrate eines Gens zu modellieren, muß man deshalb lediglich die mRNA-Konzentration betrachten. Regulatorische Einflüsse der Expression eines Gens A auf die Expression eines Gens B lassen sich vereinfachend durch eine Abhängigkeit der mRNA-Konzentration des Gens B von der mRNA-Konzentration des Gens A darstellen.
Weitere Einschränkungen ergeben sich durch die vereinfachte Betrachtung eines Genregulationsnetzwerks als ein synchrones, diskretes Zeitsystem. Daraus resultieren Probleme, wenn es bei der Durchführung von Expressionsexperimenten zu Generierung von Trainingsdaten darum geht, eine geeignete Zeitspanne Δt für die Länge eines Zustandsübergangs festzulegen. Wählt man Δt zu klein, können viele Gene ihre Expressionsrate in diesem Zeitraum nicht aktualisieren. Ist Δt dagegen zu groß, aktualisieren einige Gene in diesem Zeitraum ihre Expressionsrate mehrfach, liefern damit Informationen über indirekte Beziehungen und täuschen so regulatorische Einflüsse vor. Außerdem kommen bei großem Δt auch die Zerfallsprozesse der Genprodukte zum Tragen.
Die am Additiven Regulationsmodell orientierten Algorithmen beruhen außerdem auf der vereinfachenden, aber biologisch unrealistischen Modellannahme, das alle regulatorischen Einflüsse additiv und unabhängig voneinander wirken.
Bereits in den vorangegangenen Experimenten wurde beobachtet, wie der Reverse Engineering Prozeß durch die Meßfehler in den Daten, durch die vereinfachende Modellannahme von diskreten Expressionsraten und sicherlich auch durch algorithmenbedingte Limitationen wesentlich erschwert wurde. Da bei der Arbeit mit realen Expressionsdaten weitere vereinfachende, aber biologisch unrealistische Annahmen zum Tragen kommen, werden die einzelnen Ergebnisse der Algorithmen deshalb

schlechter ausfallen als in den vorangegangenen Simulationsexperimenten. Besonders gilt dies für die Algorithmen *REM*, *BPTT* und *Strukturlernen in kontinuierlichen DBN* sowie für den *evolutionären Algorithmus*, denn wie bereits erläutert, kamen in den vorangegangenen Experimenten an Simulationsdaten entscheidende modellbedingte Limitationen dieser Algorithmen überhaupt nicht zum Tragen.
Eine Ausnahme bildet wieder die *Adjazenzlisten-Konstruktion*. Das ihr zugrundeliegende Netzwerkmodell – ein gerichteter Graph – betrachtet lediglich die Struktur des Genregulationsnetzwerks, trifft aber keine Aussagen über dessen Dynamik. Neben Meßfehlern, die zu falschen Entscheidungen bei der Konstruktion der Erreichbarkeitsliste führen können, sind die Ergebnisse dieses Algorithmus vor allem von der Anzahl der Netzwerkkomponenten abhängig, die in einem Zyklus involviert sind. Die Ergebnisse dieses Algorithmus bei der Arbeit mit realen Expressionsdaten hängen deshalb wesentlich von den strukturellen Eigenschaften realer Genregulationsnetzwerke ab.

Kapitel 6

Anwendungsbeispiel mit realen Expressionsdaten

In diesem Kapitel werden die in der vorliegenden Arbeit behandelten Reverse Engineering Methoden nun auch an realen Expressionsdaten getestet. Dieses Anwendungsbeispiel soll dem Leser abschließend einen Eindruck davon vermitteln, inwieweit die auf abstrakten Netzwerkmodellen basierenden Reverse Engineering Methoden die Identifizierung von regulatorischen Einflüssen aus derzeit verfügbaren Expressionsdaten überhaupt ermöglichen und die Probleme bei der praktischen Anwendung der Methoden verdeutlichen.
Die Grundlage für dieses Anwendungsbeispiel bilden Experimente zur Untersuchung der intrazellulären Signaltransduktion von Interleukin-6 (IL-6) in multiplen Myelomzellen, die am Institut für klinische Immunologie und Transfusionsmedizin (IKIT) der Universität Leipzig unter der Leitung von Prof. F. Horn durchgeführt wurden [26, 50].

Biologischer Hintergrund des Anwendungsbeispiels

Multiple Myelomzellen sind entartete, unkontrolliert wachsende Plasmazellen, die das Knochenmark besiedeln und die dort stattfindende Blutbildung (Hämatopoese) unterdrücken. Diese Myelomzellen benötigen das Zytokin Interleukin-6 (IL-6) sowohl als Wachstumsfaktor als auch zum Schutz vor dem programmierten Zelltod (Apoptose). Die intrazelluläre Signaltransduktion von IL-6 verläuft zunächst über den Transkriptionsfaktor *Stat*3. Dieser wird durch Thyrosinphosphorylierung am IL-6-Rezeptor aktiviert und wandert anschließend in den Zellkern, um dort die Transkription weiterer wichtiger Transkriptionsfaktoren und bestimmter Zielgene, die das Wachstum und das Überleben der Myelomzelle sichern, direkt oder indirekt zu stimulieren (siehe Abbildung 6.1). Die Signalwege im einzelnen – also ablaufende regulatorische Interaktionen – und die daran beteiligten Gene sind noch nicht hinreichend bekannt. Ziel eines Projektes am IKIT ist es deshalb, durch die genom-

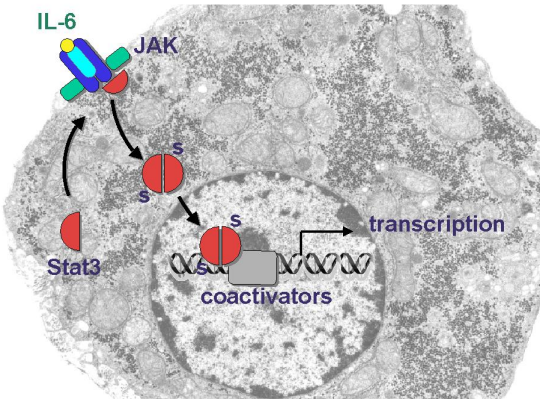


Abbildung 6.1: **Intrazelluläre Signaltransduktion von IL-6**[1]**.** Diese beginnt mit der Thyrosinphosphorylierung des Transkriptionsfaktors *Stat*3 am IL-6-Rezeptor. Der aktivierte Transkriptionsfaktor wandert anschließend in den Zellkern, um dort die Transkription bestimmter Transkriptionsfaktoren und Zielgene direkt oder indirekt zu stimulieren.

weite Analyse der Genexpressionsmuster von multiplen Myelomzellen mit Hilfe von DNA-Microarrays sowie durch gezielte Manipulation IL-6 abhängiger Signalwege und Transkriptionsfaktoren Einblicke in grundlegende regulatorische Prinzipien zu gewinnen [50].
Ergebnisse aus bisherigen Analysen der Projektgruppe und Hinweise aus der Literatur lassen auf die Existenz der in Abbildung 6.2 dargestellen regulatorischen Interaktionen schließen. So konnten mit Hilfe der durchgeführten Experimente vor allem die stimulierenden Einflüsse von *Stat*3 auf die Transkription anderer Gene nachgewiesen werden, die wichtige Transkriptionsfaktoren für die Aktivierung bestimmter Zielgene kodieren. Diese Beziehungen sind in Abbildung 6.2 durch die blauen Verbindungen beschrieben. Die Literatur liefert Hinweise sowohl für aktivierende (grüne Verbindungen) als auch für inhibitorische (rote Verbindungen) Einflüsse zwischen einzelnen Genen. Es sei darauf hingewiesen, daß Abbildung 6.2 nur die bisher experimentell aufgedeckten regulatorischen Einflüsse präsentiert. Mit hoher Wahrscheinlichkeit existieren weitere regulatorische Beziehungen zwischen den betrachteten Genen. Außerdem ist es sehr wahrscheinlich, daß die dargestellten Gene auch regulatorische Einflüsse von Genen erfahren, die in Abbildung 6.2 nicht berücksichtigt werden[2].

[1] Erhalten von Projektleiter Prof. F. Horn.
[2] Private Mitteilung von Projektleiter Prof. F. Horn.

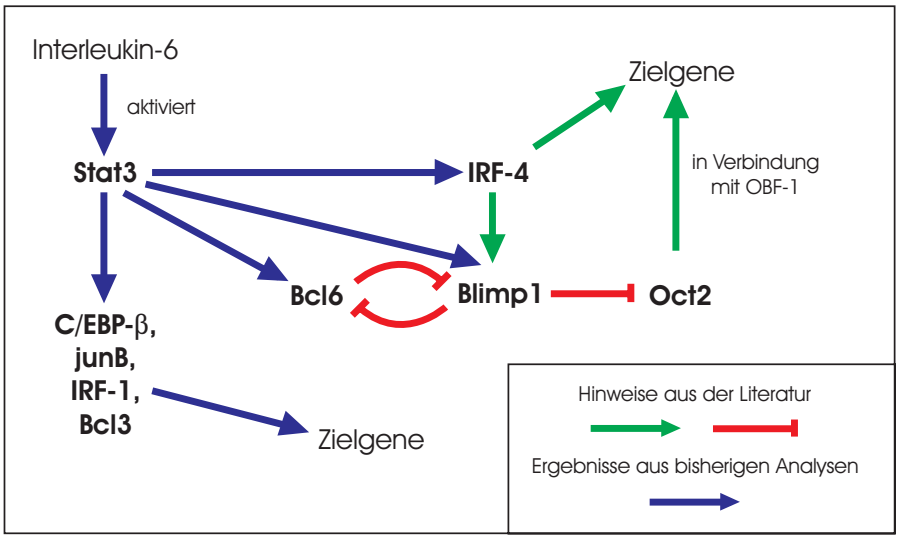


Abbildung 6.2: **Ausschnitt des Regulationsnetzwerks in Myelomzellen**[3]**.** Die hier beschriebenen regulatorischen Interaktionen ergaben sich aus den Ergebnissen bisheriger Analysen und durch Hinweise aus der Literatur. Dargestellt sind lediglich die bereits experimentell nachgewiesenen regulatorischen Interaktionen. Die Existenz weiterer regulatorischer Einflüsse ist sehr wahrscheinlich.

Experimenteller Hintergrund des Anwendungsbeispiels

Für dieses Anwendungsbeispiel standen genomweite Analysen der mRNA-Konzentrationen in Myelomzellen zur Verfügung. Alle Analysen wurden an der Myelomzellinie INA-6 durchgeführt. Wie beschrieben benötigen diese Zellen das Zytokin Interleukin-6 als Wachstumsfaktor und auch zum Schutz vor Apoptose. Da sie aufgrund dessen ohne das Zytokin auf Dauer nicht überlebensfähig sind, müssen die Zellen in einem Wachstumsmedium aufbewahrt werden, daß das Interleukin-6 enthält.

Um die Signalwege der intrazellulären Signaltransduktion zu untersuchen, wurde dem Wachstumsmedium der Zellen zunächst das Interleukin-6 für 12 Stunden entzogen und anschließend wieder zugesetzt. Die daraufhin ablaufenden Prozesse der intrazellulären Signaltransduktion von IL-6 sollten zu verschiedenen Zeitpunkten durch eine genomweite Bestimmung der Genexpressionsmuster, die den jeweiligen Systemzustand des Regulationsnetzwerks der Myelomzellen beschreiben, mit Hilfe von DNA-Microarrays protokolliert werden. Die erste Messung fand dabei zum Zeitpunkt 0h der IL-6-Restimulierung statt. Es folgten Messungen nach einer Stunde (Zeitpunkt 1h) und nach vier Stunden (Zeitpunkt 4h) der IL-6-Restimulierung. Eine weitere Messung beschreibt das genomweite Genexpressionsmuster in Myelomzellen, denen permanent Interleukin-6 zur Verfügung stand. Diese Messungen charakteri-

[3]Erhalten von Projektleiter Prof. F. Horn.

Zeitpunkt	0h IL-6		1h IL-6		4h IL-6		permanent IL-6	
	Mittelwert	Standard-abweichung	Mittelwert	Standard-abweichung	Mittelwert	Standard-abweichung	Mittelwert	Standard-abweichung
Bcl3	225,78	75,66	367,37	52,77	361,77	109,09	310,57	59,91
Bcl6	25,83	4,39	81,33	40,03	49,50	16,22	39,70	15,55
Blimp1	65,90	44,66	223,97	41,60	34,03	22,27	38,07	31,16
C/EBP-β	338,28	138,72	858,23	30,86	615,63	254,86	394,13	185,18
IRF- 1	175,75	72,37	310,77	40,01	146,07	84,49	126,20	37,99
IRF- 4	67,93	23,68	79,33	64,61	56,20	32,52	107,70	52,06
JunB	97,90	24,94	660,40	159,69	166,75	69,47	137,02	47,35
Oct 2	112,70	56,44	123,82	49,46	316,05	136,34	253,45	86,24
Stat3	186,99	74,02	237,67	73,34	402,72	174,76	350,82	122,94

Tabelle 6.1: **Abweichungen in wiederholten Messungen der Expressionsraten zu einem Zeitpunkt.** Die Tabelle gibt für jeden der vier gemessenen Zeitpunkte den jeweiligen Mittelwert und die zugehörigen Standardabweichungen der Expressionsraten aller in Abbildung 6.2 betrachteten neun Gene an.

sieren somit das zeitliche Verhalten des Genregulationsnetzwerks der Myelomzellen an den vier aufeinanderfolgenden Zeitpunkten 0h, 1h, 4h und permanent.
Jede Messung wurde mehrfach durchgeführt. Als Resultat entstanden vier Messungen des genomweiten Genexpressionsmusters zum Zeitpunkt 0h und jeweils drei Messungen des genomweiten Genexpressionsmusters zu den übrigen Zeitpunkten. Weil bei der Extraktion der mRNA die Zellen zerstört werden, stammt jede der insgesamt 13 Messungen von einer anderen Zellpopulation. Das genomweite Expressionsverhalten zweier Zellen zu einem bestimmten Zeitpunkt kann sich durchaus voneinander unterscheiden. Da man aber mit den Durchschnittswerten aus einer Zellpopulation arbeitet, darf man annehmen, daß alle Zellpopulationen zum Zeitpunkt 0h das gleiche durchschnittliche Expressionsmuster aufwiesen. Die Messungen zu einem Zeitpunkt können deshalb als echte Replikate und die Messungen zu späteren Zeitpunkten als Folgezustände der Messungen zu vorangegangenen Zeitpunkten betrachtet werden.
Die einzelnen durchschnittlichen Expressionsraten der Gene in den wiederholten Messungen zu einem Zeitpunkt können stark schwanken. Um dies zu verdeutlichen, sind in Tabelle 6.1 für jeden Zeitpunkt die Mittelwerte und die Standardabweichungen der Expressionsraten aller bereits in Abbildung 6.2 betrachten neun Gene aufgelistet. Gründe hierfür sind zum einen Meßfehler und zum anderen natürlich auch der Sachverhalt, daß die jeweiligen Messungen zu einem Zeitpunkt an verschiedenen Zellpopulationen durchgeführt wurden. Wie zu erkennen, unterliegen größere Meßwerte im allgemeinen auch größeren Schwankungen. Um bei der Analyse der Expressionsdaten falsche Schlüsse, die aufgrund von Meßfehlern, Ausreißerwerten und Inkonsistenzen entstehen können, zu vermeiden, sollten die Meßwerte zu einem Zeitpunkt gemittelt werden. Danach steht für die Analyse der Daten aber lediglich eine Zeitreihe aus den vier aufeinanderfolgenden, gemittelten Systemzuständen zur Verfügung. Für die Identifizierung regulatorischer Interaktionen mit Hilfe von Reverse Enginee-

ring Methoden ist dieser Datenumfang jedoch viel zu klein. Die Untersuchungen in diesem Abschnitt arbeiten deshalb trotz der großen Unterschiede mit den Werten aus den einzelnen Messungen. Jede Kombination einer beliebigen Messung zu einem bestimmten Zeitpunkt mit einer beliebigen Messung zum darauffolgenden Zeitpunkt wird dann als ein Zustandsübergangspaar angesehen. Insgesamt lassen sich so 30 Zustandsübergänge generieren. Hierbei muß allerdings berücksichtigt werden, daß diese 30 Zustandsübergänge letztendlich auch nur wiederholte Beobachtungen der drei Zustandsübergänge 0h-1h, 1h-4h, 4h-permanent darstellen. Im vorangegangenen Abschnitt wurden bei der Arbeit mit Zeitreihen die Probleme deutlich, die sich bei der wiederholten Betrachtung bestimmter Systemzustände ergeben. Weiterhin ist auch klar, daß durch eine solche Betrachtung der Daten Inkonsistenzen entstehen – bei der Generierung der Zustandsübergangsdaten werden einem Anfangszustand verschiedene, zum Teil recht unterschiedliche Folgezustände zugeordnet. Aufgrund der Begrenzung des verfügbaren Datenumfangs ist eine solche Betrachtungsweise der Daten jedoch erforderlich.

Anwendung der Reverse Engineering Methoden

Das Ziel ist es, zu untersuchen, welche der in Abbildung 6.2 beschriebenen regulatorischen Einflüsse bei der Analyse der gegebenen Expressionsdaten mit Hilfe der in dieser Arbeit vorgestellten Reverse Engineering Methoden identifiziert werden können. Mit Ausnahme der *Adjazenzlisten-Konstruktion* wurden alle in Tabelle 5.1 aufgelisteten Algorithmen an den realen Expressionsdaten getestet. Dieser Algorithmus erwartet als Eingabe stabile Zustandsdaten, die die Ergebnisse aus gezielten Manipulationsexperimenten enthalten. Solche Daten standen leider nicht zur Verfügung.
Die für die Messung der Genexpressionsmuster verwendeten Affymetrix U95A Oligonukleotid-Arrays enthalten 12000 spezifische Proben menschlicher Gene. Eine Analyse der gemessenen Genexpressionsmuster konnte 104 differentiell exprimierte Gene aufdecken. Diese Gene wiesen also in den Messungen der Genexpressionsmuster zu den vier Zeitpunkten 0h, 1h, 4h und permanent ein sich änderndes Expressionsverhalten auf und sind damit sehr wahrscheinlich in irgendeiner Form an der intrazellulären Signaltransduktion von IL-6 beteiligt. Um alle 104 differentiell exprimierten Gene im Reverse Engineering Prozeß zu berücksichtigen, liefern die aus den gegebenen Expressionsdaten konstruierten 30 Zustandsübergänge zu wenige Informationen. Daher wurde das Reverse Engineering auf die in Abbildung 6.2 betrachteten neun Gene $Bcl3$, $Bcl6$, $Blimp1$, $C/EBP\text{-}\beta$, $IRF\text{-}1$, $IRF\text{-}4$, $junB$, $Oct2$ und $Stat3$ beschränkt.
Die Algorithmen arbeiteten mit den generierten 30 Zustandsübergangspaaren. Da der Algorithmus *BPTT* als Eingabe explizit eine Zeitreihe erwartet, wurden ihm die gemittelten Messungen der Genexpressionsmuster zu den vier betrachteten, aufeinanderfolgenden Zeitpunkten für das Reverse Engineering übergeben.

Folgende Anpassungen der Reverse Engineering Algorithmen wurden vorgenommen: Die Signifikanzniveaus α_1 und α_2 für den Algorithmus *Reveal* bekamen die Werte $\alpha_1 = 0.01$ und $\alpha_2 = 0.00001$ zugewiesen. Es wurde also ein sehr kleiner Wert für α_2 ausgewählt, da die Experimente an Simulationsdaten gezeigt haben, daß sich vor allem die Irrtumswahrscheinlichkeit $\alpha_{2,gesamt}$ schwer kontrollieren läßt und zu einer großen Anzahl falsch identifizierter Einflüsse führen kann.
Die gemessenen Expressionsraten sind wesentlich höher als die Werte der Expressionsraten in den Simulationsdaten. Daraus resultierend ergeben sich betragsmäßig kleinere Schätzer der Gewichte w_{ij}. Der Schwellwert *threshold*, den der *evolutionäre Algorithmus* für die Klassifizierung der Schätzer $\hat{w}_{ij}$ nutzt, wurde deshalb ebenfalls kleiner gewählt. Anstelle von $threshold = 0.5$ in den Experimenten mit Simulationsdaten verwendete der Algorithmus hier den Wert 0.2.
Wie bereits die vorangegangen Experimenten gezeigt haben, benötigt der Algorithmus *Strukturlernen in diskreten DBN* einen großen Datenumfang. Mit den hier verfügbaren Daten gelingt es dem Algorithmus nicht, Zusammenhänge zu identifizieren. Um den von ihm benötigten Datenumfang etwas zu verringern, wurden hier deshalb bei der Diskretisierung der Expressionsdaten die kontinuierlichen Werte nicht auf die drei diskreten Zustände „−1“, „0“ und „1“ abgebildet, sondern es wurde analog zu den Booleschen Netzwerken nur mit den zwei Zuständen „0“ und „1“ gearbeitet.

Diskussion der Ergebnisse

Die Tabelle 6.2 gibt einen Überblick, welche der insgesamt elf bereits experimentell nachgewiesenen regulatorischen Einflüsse aus Abbildung 6.2 von den einzelnen Reverse Engineering Methoden identifiziert werden konnten.
Natürlich erscheinen die Ergebnisse zunächst recht unbefriedigend. Die besten Ergebnisse erzielte der Algorithmus *Reveal* – ihm gelang der Nachweis von sieben der elf bekannten regulatorischen Einflüsse. Alle andere Algorithmen konnten lediglich vier (*Strukturlernen in kontinuierlichen DBN*, *REM* – linearen Ansatz), drei (*BPTT*, *evolutionärer Algorithmus*) bzw. zwei (*REM* – nichtlinearer Ansatz, *Strukturlernen in diskreten DBN*) dieser bereits nachgewiesenen Einflüsse identifizieren.
Bei der Betrachtung der Ergebnisse muß allerdings folgender Sachverhalt unbedingt berücksichtigt werden: Die Algorithmen können nur solche Einflüsse aufdecken, über die die gegebenen Daten Informationen liefern. Auch wenn eine verfügbare Anzahl von 30 Zustandsübergängen recht groß erscheint, so beschreiben diese doch trotzdem nur drei verschiedene Zustandsübergänge des Systems und liefern sehr wenige Informationen über sein dynamisches Verhalten. Inkonsistenzen, die wie erläutert bei der Generierung der Zustandsübergangspaare entstehen, erschweren die Identifizierung regulatorischer Einflüsse zusätzlich.
Das Diagramm 6.3 beschreibt den zeitlichen Verlauf des Expressionsverhaltens der einzelnen Gene mit Hilfe der gemittelten Messungen der Expressionsraten zu den

Regulatorische Einflüsse	Reveal	Struktur-lernen diskrete DBN	REM (linear)	REM (nichtlinear)	evolutionärer Algorithmus	BPTT	Struktur-lernen kontinuier-liche DBN
Stat3 → IRF-4						x	x
Stat3 → Blimp1	x						x
Stat3 → Bcl6	x						
Stat3 → C/EBP-β							
Stat3 → junB	x			x			x
Stat3 → IRF-1	x		x				
Stat3 → Bcl3							
IRF-4 → Blimp1			x		x	x	x
Bcl6 → Blimp1	x		x		x		
Blimp1 → Bcl6	x	x					
Blimp1 → Oct2	x	x	x	x	x	x	
# truePos	7	2	4	2	3	3	4
# zusätzlich identifizierte Einflüsse	15	6	24	23	13	20	15

Tabelle 6.2: **Ergebnisse der Reverse Engineering Methoden bei der Arbeit mit realen Expressionsdaten.** Das Zeichen „x"gibt an, daß der jeweilige Algorithmus den entsprechenden regulatorischen Einfluß identifizieren konnte. Zusätzlich identifizierte Einflüsse müssen nicht unbedingt falsch identifizierte Einflüsse sein, da weitere Interaktionen zwischen den betrachteten Genen nicht ausgeschlossen werden können.

jeweiligen Zeitpunkten. Es faßt damit die Informationen zusammen, die die gegebenen Daten liefern und ist bei einer detaillierten Analyse der Ergebnisse hilfreich. Bei der genaueren Betrachtung der Ergebnisse fällt vor allem folgendes auf: Besonders schwer scheint der Nachweis des aktivierenden Einflusses von $Stat3$ auf die Transkription anderer Gene. Ein Blick auf das Diagramm 6.3 zeigt, daß diese Beziehungen durch die Daten auch nicht beschrieben werden: Während für die von $Stat3$ regulierten Gene ein starker Anstieg der Expressionsrate bereits zum Zeitpunkt 1h zu erkennen ist, kann für $Stat3$ selbst erst zum Zeitpunkt 4h eine starke Zunahme der Expressionsrate beobachtet werden. Die Daten liefern also keine Informationen über den aktivierenden Einfluß von $Stat3$ als Ursache für die erhöhten Expressionsraten der Gene $Blimp1$, $Bcl6$, $C/EPB\text{-}\beta$, $junB$, $IRF\text{-}1$ sowie $Bcl3$, und die Algorithmen können die entsprechenden regulatorischen Einflüsse damit auch nicht nachweisen. Wenn die intrazelluläre Signaltransduktion von IL-6 aber mit der Aktivierung von $Stat3$ beginnt, warum spiegelt sich dies nicht in den Daten wider? An dieser Stelle kommt die vereinfachte Betrachtung der Genregulationsprozesse zum Tragen. Wie bereits mehrfach beschrieben, werden die Genregulationsmechanismen auf die Transkription eingeschränkt und alle Mechanismen auf anderen

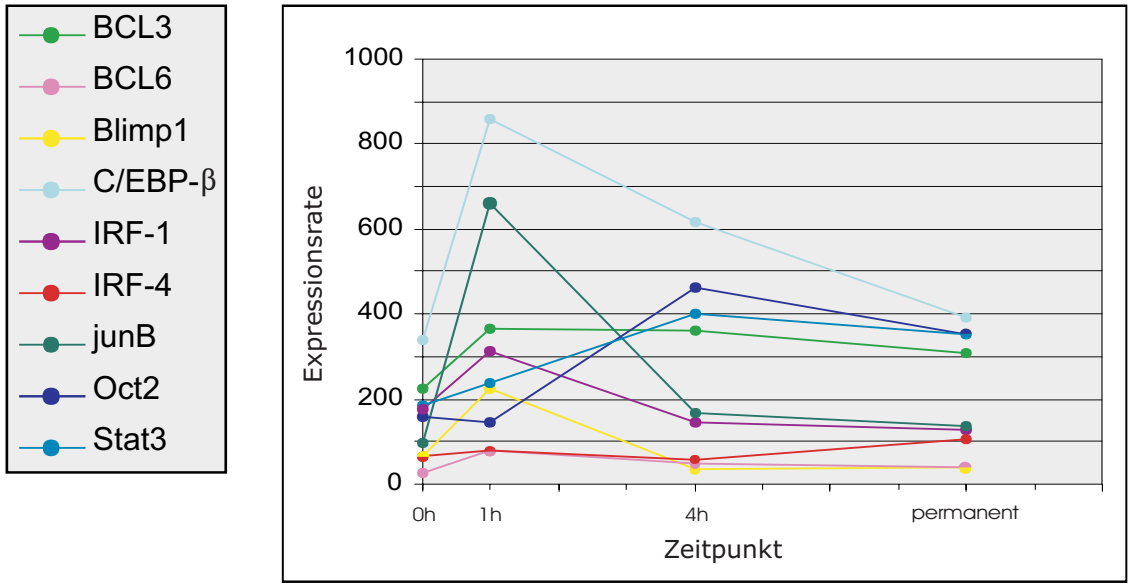


Abbildung 6.3: **Zeitlicher Verlauf des Expressionsverhaltens der einzelnen Gene.** Zu sehen sind die gemittelten Messungen der Expressionsraten zu den jeweiligen Zeitpunkten.

Ebenen der Genexpression ignoriert. Man nimmt dann eine starke Korrelation zwischen den mRNA-Konzentrationen und den Protein-Konzentrationen der Gene an und beschränkt sich bei der Charakterisierung der Expressionsrate eines Gens auf die zugehörige mRNA-Konzentration. Das daraus resultierende Problem bei dieser konkreten Anwendung ist in Abbildung 6.4 graphisch veranschaulicht: Die Zugabe von IL-6 aktiviert das von *Stat*3 kodierte Protein. Dies hat keine Auswirkungen auf die mRNA-Konzentration von *Stat*3! Der aktivierte Transkriptionsfaktor stimuliert dann die Transkription von *IRF*-4, *Blimp*1, *Bcl*6, *C/EPB-β*, *junB*, *IRF*-1 sowie *Bcl*3, und deren mRNA-Konzentration steigt an. Anhand der mRNA-Konzentrationen allein ist der aktivierende Einfluß von *Stat*3 hier also nicht nachweisbar. Um diesen Nachweis zu ermöglichen, müßten die Protein-Konzentrationen in die Analyse einbezogen werden!
Warum gelingt es dann den Algorithmen dennoch, einige dieser von *Stat*3 ausgehenden regulatorischen Einflüsse nachzuweisen? Während für die mRNA-Konzentration von *Stat*3 zum Zeitpunkt 4h eine starke Zunahme beobachtet werden kann, sind die Expressionsraten der Gene *Blimp*, *junB* und *IRF*-1 zu diesem Zeitpunkt bereits wieder stark abgefallen. Dies läßt die Algorithmen fälschlicherweise einen repressorischen Einfluß von *Stat*3 auf diese Gene vermuten. Eine Analyse der von den Algorithmen *REM*, *BPTT* und *Strukturlernen in kontinuierlichen DBN* jeweils geschätzten Gewichten konnte bestätigen, daß diese ein negatives Vorzeichen aufwiesen. Eine Ausnahme stellt der regulatorische Einfluß *Stat*3 → *IRF*-4 dar. Ein leichter Anstieg der Expressionsrate von *IRF*-4 ist erst zu beobachten, nachdem sich

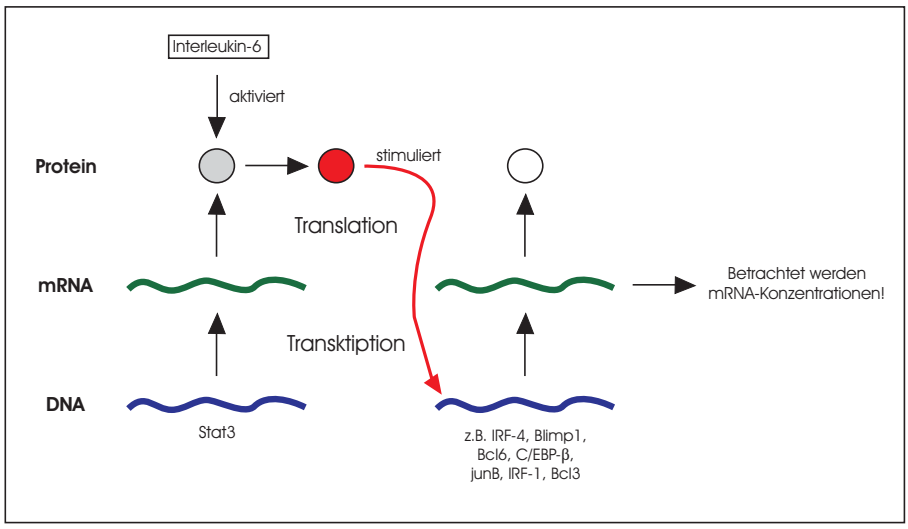


Abbildung 6.4: **Ablaufende Prozesse bei der Signaltransduktion von IL-6.** Das Zytokin IL-6 aktiviert das von *Stat*3 kodierte Protein. Dies hat keine Auswirkungen auf die mRNA-Konzentration von *Stat*3. Der aktivierte Transkriptionsfaktor stimuliert dann die Transkription von IRF-4, *Blimp*1, *Bcl*6, C/EPB-β, $junB$, IRF-1 sowie *Bcl*3, und deren mRNA-Konzentrationen steigen an. Da nur die mRNA-Konzentrationen betrachtet werden, läßt sich ein aktivierender Einfluß von *Stat*3 nicht als Ursache für die erhöhte Transkription der anderen Gene nachweisen!

zum Zeitpunkt 4h die Expressionsrate von *Stat*3 stark erhöht hatte. Daraus schließen die Algorithmen *BPTT* und *Strukturlernen in kontinuierlichen DBN* korrekt auf einen aktivierenden Einfluß von *Stat*3 auf IRF-4.
Der Nachweis der übrigen vier Einflüsse IRF-4 $\rightarrow$ *Blimp*1, *Blimp*1 $\rightarrow$ *Oct*2, *Blimp*1 $\rightarrow$ *Bcl*6 und *Bcl*6 $\rightarrow$ *Blimp*1 gelingt den Algorithmen etwas besser. Aufgrund der wenigen verfügbaren Information können die einzelnen Algorithmen aber auch hier nicht jeden der Zusammenhänge identifizieren. Besonders einfach scheint der Nachweis von *Blimp*1 $\rightarrow$ *Oct*2 – fast alle Algorithmen konnten diesen Einfluß aufdecken. Ein Blick in das Diagramm 6.3 bestätigt, daß die gegebenen Daten diesen Einfluß auch sehr gut unterstützen. Außerdem erfährt das Gen *Oct*2 – zumindest nach den bisherigen Erkenntnissen in Abbildung 6.2 – nur diesen einen regulatorischen Einfluß. Schon die vorangegangenen Simulationsexperimente konnten zeigen, daß sich solche einzelnen regulatorischen Einflüsse wesentlich besser nachweisen lassen als zusammengesetzte Einflüsse.
Der regulatorische Einfluß IRF-4 $\rightarrow$ *Bimp*1 kann zumindest von den mit kontinuierlichen Expressionsraten arbeitenden Algorithmen gut identifiziert werden. Den mit diskreten Expressionsraten arbeitenden Algorithmen gelingt dieser Nachweis hingegen nicht. Der Grund hierfür liegt in der Diskretisierung der Expressionsraten. Dabei gehen wichtige Informationen über die recht kleinen Schwankungen der Expressionsrate von Gen IRf-4 verloren. Diese erscheint deshalb im diskreten Raum

als eine konstante Größe und ist nicht als Ursache für das Expressionsverhalten von *Blimp*1 nachweisbar.
Im Gegensatz dazu konnten die mit diskreten Expressionsraten arbeitenden Algorithmen die Zusammenhänge zwischen den Genen *BCL*6 und *Blimp*1 recht gut aufdecken, denn die in den Daten enthaltenen Informationen über diese Zusammenhänge werden bei der Diskretisierung nicht zerstört. Den übrigen Algorithmen gelingt der Nachweis dieser zwei regulatorischen Einflüsse dagegen kaum.

Weiterhin ist in Tabelle 6.2 zu erkennen, daß die Algorithmen neben den wenigen, korrekt identifizierten regulatorischen Einflüssen auch eine recht hohe Anzahl zusätzlich identifizierter Einflüsse liefern, bei denen es sich sicherlich oftmals um falsch identifizierte Zusammenhänge handelt. Hierfür gibt es mehrere Gründe:
Für die von *Stat*3 regulierten Gene ist ein recht ähnliches Expressionsverhalten in den Expressionsdaten zu beobachten. Da *Stat*3 selbst aus den geschilderten Gründen nicht als zentrales Regulatorgen zu identifizieren ist, erklären die Algorithmen das ähnliche Expressionsverhalten der von ihm regulierten Gene durch regulatorische Einflüsse zwischen diesen.
Ferner erschwert das ähnliche Expressionsverhalten der Gene eine korrekte Selektion der tatsächlich existierenden regulatorischen Einflüsse. Als Beispiel betrachte man die Gene *Blimp*1, *junB* und *IRF*-1. Das Gen *Blimp*1 übt einen regulatorischen Einfluß auf das Gen *Oct*2 aus, der von den gegebenen Daten gut beschrieben wird. Aufgrund des ähnlichen Expressionsverhaltens von *Blimp*1, *junB* und *IRF*-1 sind prinzipiell aber auch die regulatorischen Einflüsse $junB \rightarrow Oct2$ und $IRF\text{-}1 \rightarrow Oct2$ möglich. Warum sollen die Algorithmen also richtig entscheiden können, daß das Gen *Oct*2 von *Blimp*1 reguliert wird und nicht von *junB* oder *IRF*-1? Es sind nur wenige Daten vorhanden, die den Algorithmen nicht genügend Informationen liefern, um zwischen korrekten und falschen regulatorischen Einflüsse zu differenzieren.
Weiterhin kann auch die wiederholte Beobachtung der Zustandsübergänge in den 30 Zustandsübergangspaaren einen Grund für die Identifizierung falscher Einflüsse liefern. Wie in den vorangegangenen Experimenten deutlich wurde, kann die wiederholte Beobachtung bestimmter Zustandsübergänge die Entscheidung für einen falschen Zusammenhang zusätzlich unterstützen.
Schließlich müssen neben den bekannten regulatorischen Zusammenhängen weitere regulatorische Einflüsse existieren. Da zum Beispiel die Expressionsraten bestimmter Gene (*Bcl*3, $C/EBP\text{-}\beta$ und *IRF*-1) nach dem Zeitpunkt 1h wieder abnehmen, obwohl die Expressionsrate von *Stat*3 sogar zunimmt und auch Interleukin-6 für die Aktivierung von *Stat*3 vorhanden ist, kann der aktivierende Einfluß von *Stat*3 allein das Expressionsverhalten dieser Gene nicht erklären. Weitere regulatorische Einflüsse können zum einen von Genen ausgehen, die im Reverse Engineering Prozeß nicht betrachtet wurden. Dann entstehen Probleme vor allem, wenn es sich bei einem nicht berücksichtigten Gen um ein wichtiges Regulatorgen handelt, daß regulatorische Einflüsse auf mehrere der betrachteten Gene ausübt und ein ähnliches

sehr gut möglich	denkbar, aber bisher keine Hinweise	eher unwahrscheinlich
Bcl3 → junB	Bcl6→ Oct2	Bcl6 → IRF1
Oct2→ Blimp1	IRF1 → IRF1	Blimp1 → IRF1
Stat3→ Oct2	junB→ IRF4	IRF1→ Blimp1
	junB → Oct2	IRF1 → junB
	junB → Stat3	IRF-4 → IRF-1
	junB → junB	IRF-4 → Oct2
	junB→ Blimp1	IRF-4 → Stat3
	Oct2→ C/EBP-β	Oct2→ IRF-1

Tabelle 6.3: **Bewertung zusätzlich identifizierter Einflüsse.** Zusätzlich identifizierte Zusammenhänge, die mindestens drei der sieben Algorithmen nachgewiesen haben, wurden von Projektleiter Prof. F. Horn bezüglich der Fragestellung bewertet, ob ihre Existenz prinzipiell denkbar oder eher unwahrscheinlich ist.

Expressionsverhalten dieser bewirkt. Wie schon für das zentrale Regulatorgen Stat3[4] erläutert, vermuten die Algorithmen deshalb regulatorische Einflüsse zwischen den regulierten Genen, da sie die tatsächlichen Einflüsse als Ursache für das Expressionsverhalten der Gene nicht nachweisen können. Zum anderen kann es weitere Beziehungen auch zwischen den betrachteten Genen selbst geben, und die von den Algorithmen zusätzlich identifizierten Einflüsse müssen nicht immer falsch sein. Diese Einflüsse wurden deshalb näher analysiert. In Tabelle 6.3 sind all die Einflüsse aufgelistet, die mindestens drei der sieben Algorithmen zusätzlich identifizierten. Jeder einzelne Zusammenhang wurde durch den Projektleiter Prof. F. Horn bezüglich der Fragestellung bewertet, ob seine Existenz prinzipiell denkbar oder eher unwahrscheinlich ist. Wie zu erkennen, können viele der zusätzlich identifizierten Einflüsse zumindest nicht ausgeschlossen werden. Einige sind sogar sehr wahrscheinlich.

Was läßt sich nun über den Vergleich der Algorithmen sagen? Läßt man wie in den vorangegangenen Simulationsexperimenten die Qualität eines identifizierten regulatorischen Einflusses (aktivierender oder inhibitorischer Einfluß) bei der Bewertung der Güte eines rekonstruierten Netzwerks außer acht, dann können die Ergebnisse aus diesem Experiment die Beobachtungen aus den vorangegangenen Experimenten an Simulationsdaten im allgemeinen bestätigen (siehe Abschnitt 5.5). Der Algorithmus *Strukturlernen in kontinuierlichen DBN* liefert auch hier bessere Ergebnisse als die auf dem nichtlinearen Additiven Regulationsmodell basierenden Algorithmen. Der *evolutionäre Algorithmus* und der Algorithmus *BPTT* können beide drei der bereits bekannten regulatorischen Einflüsse identifizieren. Allerdings liefert der Algorithmus *BPTT* eine größere Anzahl an zusätzlich identifizierten Einflüssen. Das

[4]Das Gen *Stat*3 wurde zwar im Reverse Engineering Prozeß betrachtet, die von ihm ausgehenden regulatorischen Einflüsse sind aber – wie oben diskutiert – aufgrund der vereinfachten Betrachtungsweise der Genregulationsprozesse nicht nachweisbar.

entspricht den Ergebnissen aus der Arbeit mit Simulationsdaten, bei der dieser Algorithmus oftmals einen kleineren positiv prädiktiven Wert erzielte als der *evolutionäre Algorithmus*. Wieder müssen für den Algorithmus *REM* (nichtlinearer Ansatz) die unbefriedigendsten Ergebnisse beobachtetet werden.
Der Algorithmus *Reveal* konnte in diesem Experiment die besten Ergebnisse liefern. Schon bei der Arbeit mit Simulationsdaten wurde deutlich, daß dieser Algorithmus im Gegensatz zu den anderen Algorithmen in der Lage ist, wiederholte Beobachtungen in den Daten für die Identifizierung regulatorischer Einflüsse zu nutzen. Der Algorithmus *Strukturlernen in diskreten DBN* benötigt dagegen für die Identifizierung regulatorischer Einflüsse sehr viel mehr Informationen, und so konnte er hier lediglich zwei bekannte Einflüsse nachweisen. Im Unterschied zu diesen beiden Algorithmen liefert der Algorithmus *REM* (linearer Ansatz) vergleichsweise vor allem eine recht hohe Anzahl zusätzlicher regulatorischer Einflüsse.
In diesem Experiment können jetzt die Ergebnisse aller Algorithmen miteinander verglichen werden: Es entfällt die Überlegenheit der auf dem nichtlinearen Additiven Regulationsmodell bzw. auf dem kontinuierlichen DBN basierenden Algorithmen, die sich für diese Algorithmen bei der Arbeit mit Simulationsdaten aufgrund der Tatsache ergab, daß die Generierung der Simulationsdaten ebenfalls auf dem nichtlinearen Additiven Regulationsmodell beruhte. Die Ergebnisse dieser Algorithmen fallen deshalb in diesem Experiment nicht besser aus als die der anderen Algorithmen. Wie beschrieben, konnte der Algorithmus *Reveal* sogar bessere Ergebnisse erzielen. Zum einen kommt der negative Einfluß der vereinfachten Modellannahme diskreter Expressionsraten in dieser konkreten Anwendung kaum zum Tragen. Zwar gehen bei der Diskretisierung der Daten durchaus Informationen verloren. Deshalb konnte der Algorithmus *Reveal* zum Beispiel den regulatorischen Einfluß $IRF\text{-}4 \rightarrow Bimp1$ nicht identifizieren. Die meisten Gene zeigen aber recht deutliche Veränderungen ihres Expressionsverhaltens, die auch bei der Diskretisierung der Daten erhalten bleiben. Damit werden wichtige Informationen über regulatorische Zusammenhänge durch die Diskretisierung nicht zerstört. Zum anderen ergeben sich für die am Additiven Regulationsmodell orientierten Algorithmen aufgrund der abstrakten Betrachtungsweise der Genregulationsprozesse im allgemeinen größere Probleme als für den Algorithmus *Reveal*. Während dieser auf einer qualitativen Ebene arbeitet und lediglich hohe und niedrige Expressionsraten miteinander in Beziehung setzt, arbeiten die am Additiven Regulationsmodell orientierten Algorithmen mit kontinuierlichen Werten. Sie versuchen zu erklären, wie sich die mRNA-Konzentrationen eines Gens aus den mRNA-Konzentrationen der dieses Gen regulierenden Gene errechnet. Da die mRNA-Konzentration eines Gens aber eigentlich von den Protein-Konzentrationen der dieses Gen regulierenden Gene abhängig ist, erfordert dieses Verfahren mindestens einen linearen Zusammenhang zwischen den mRNA-Konzentrationen und den Protein-Konzentrationen der Gene. Dieser ist in der Praxis oftmals nicht gegeben, was die korrekte Identifizierung regulatorischer Einflüsse für diese Algorithmen

erheblich erschwert[5].
Schließlich muß für dieses Anwendungsbeispiel auch festgestellt werden, daß der Algorithmus *REM* bei der Arbeit mit dem linearen Additiven Regulationsmodell vier der bekannten regulatorischen Einflüsse identifizieren konnte, während er bei der Arbeit mit dem nichtlinearen Additiven Regulationsmodell lediglich zwei dieser Einflüsse nachwies. Es scheint also, daß die gegebenen Daten durch den linearen Ansatz besser erklärt werden.

Zusammenfassung

Die Ergebnisse aus diesem Anwendungsbeispiel sind auf den ersten Blick recht unbefriedigend. Nur wenige der bekannten Einflüsse konnten identifiziert werden. Daneben lieferten die Algorithmen eine recht hohe Anzahl zusätzlicher Einflüsse.
Sicherlich liegt dies auch an den Reverse Engineering Methoden sowie ihren modell- und algorithmenbedingten Limitationen selbst. So konnten vor allem die von $Stat3$ ausgehenden, stimulierenden regulatorischen Einflüsse aufgrund der abstrakten Betrachtung der Genregulationsprozesse kaum nachgewiesen werden. Weiterhin wurde am Beispiel des regulatorischen Einflusses $IRF\text{-}4 \rightarrow Bimp1$ deutlich, daß durch die Diskretisierung der Daten wichtige Informationen verloren gehen können und regulatorische Zusammenhänge dann nicht mehr nachweisbar sind. Für die auf dem Additiven Regulationsmodell bzw. auf dem kontinuierlichen DBN basierenden Algorithmen erschwerte sicherlich auch die Annahme der Additivität und Unabhängigkeit regulatorischer Einflüsse das Reverse Engineering. Die Algorithmen *BTPP*, *Strukturlernen in DBN* sowie der *evolutionäre Algorithmus* können außerdem in einem lokalen Optimum stecken bleiben.
Der Hauptgrund für die weniger zufriedenstellenden Ergebnisse ergibt sich aber aus der Begrenzung der verfügbaren Daten. Die Algorithmen können nur die Informationen interpretieren, die ihnen die Daten liefern. Zum einen war der Umfang der für dieses Anwendungsbeispiel gegebenen Daten sehr gering. Wie bereits die vorangegangenen Experimente mit Simulationsdaten gezeigt haben, reicht ein derart beschränkter Datenumfang nicht aus, um die bestehenden regulatorischen Einflüsse korrekt zu identifizieren. Außerdem wurden im Reverse Engineering Prozeß nur die neun Gene betrachtet, die an bereits bekannten regulatorischen Zusammenhängen beteiligt sind. Wichtige Regulatorgene werden deshalb im Reverse Engineering Prozeß vielleicht nicht berücksichtigt, was zu den beschriebenen Problemen führen kann. Wäre ein größerer Datenumfang verfügbar, könnte man alle 104 differentiell exprimierten Gene in den Reverse Engineering Prozeß integrieren und so die Wahrscheinlichkeit reduzieren, ein wichtiges Regulatorgen versehentlich nicht zu betrachten.
Zum anderen sind die verfügbaren Daten auch hinsichtlich ihres Typs begrenzt. Die Messungen der Protein-Konzentrationen sind sehr ungenau und viel aufwendiger als die Messungen der mRNA-Konzentrationen. Sie stehen deshalb nicht in dem

[5]Vergleiche Unterabschnitt *Limitationen* in 3.4.1.

gleichen Umfang zur Verfügung, wie die mRNA-Konzentrationen. Dadurch wird die vereinfachte Betrachtung der Genregulationsprozesse überhaupt erst notwendig. Die beschriebenen, daraus resultierenden Probleme sind also weniger Limitationen der Reverse Engineering Methoden, sondern vielmehr Limitationen der verfügbaren Daten!

Kapitel 7

Diskussion

7.1 Zusammenfassung

Neue Technologien auf dem Gebiet der Molekularbiologie liefern eine Fülle von Daten über das Expressionsverhalten mehrerer tausend Gene, deren Auswertung den Einsatz geeigneter Analyse- und Modellierungsmethoden erfordert. Einen wichtigen Ansatz hierbei bilden Reverse Engineering Methoden die versuchen, regulatorische Interaktionen zwischen den Genen aufzudecken und mit der Rekonstruktion des zugrundeliegenden genetischen Netzwerks das komplexe Zusammenspiel der Gene zu verstehen. Das Ziel dieser Arbeit war es, einen umfassenden Überblick über diesen Ansatz der Datenanalyse zu vermitteln:

Um dem Leser eine praktische Anwendung der Reverse Engineering Methoden zu erleichtern und ihm beim Verständnis ausgewählter Ansätze zu helfen, wurden zunächst die theoretischen Aspekte des Reverse Engineerings betrachtet. Im allgemeinen läßt sich der Reverse Engineering Prozeß in zwei Teilschritte untergliedern: Als erstes ist unter Berücksichtigung der verfügbaren Expressionsdaten und der gegebenen Fragestellung ein genetisches Netzwerkmodell festzulegen, daß zur Beschreibung der Genexpressions- und Genregulationsprozesse dienen soll. Aufgrund der Begrenzung derzeit verfügbarer Expressionsdaten muß hierbei stark von der biologischen Realität abstrahiert werden. Man ignoriert vereinfachend, daß Regulationsprozesse neben der Transkription auch auf anderen Ebenen der Genexpression stattfinden und nimmt so eine starke Korrelation zwischen den mRNA-Konzentrationen und den Protein-Konzentrationen der Gene an. Die Expressionsraten der Gene können dann allein durch die zugehörigen mRNA-Konzentrationen beschrieben werden. Als Beispiele für solche abstrakten Netzwerkmodelle wurden gerichtete Graphen, Boolesche Netzwerke, diskrete und kontinuierliche Dynamische Bayessche Netzwerke sowie Additive Regulationsmodelle eingeführt.
Im zweiten Teilschritt des Reverse Engineering Prozesses wird dann ein geeigneter Reverse Engineering Algorithmus benötigt, der die Parameter des ausgewählten

Netzwerkmodells mit Hilfe der gegebenen Expressionsdaten bestimmt und festlegt, zwischen welchen Komponenten des Netzwerks regulatorische Einflüsse bestehen. Ein geeigneter Reverse Engineering Algorithmus für jedes der vorgestellten Netzwerkmodelle wurde detailliert beschrieben sowie seine modell- und algorithmusbedingten Limitationen analysiert: Als ein möglicher Reverse Engineering Algorithmus für gerichtete Graphen bietet sich die *Adjazenzlisten-Konstruktion* an. Einen bekannten Ansatz für das Reverse Engineering in Booleschen Netzwerken stellt der Algorithmus *Reveal* dar. Für die Additiven Regulationsmodelle wurden die Algorithmen *Reverse Engineering in Matrizen (REM)*, *Backpropagation through time (BPTT)* und ein *evolutionärer Algorithmus* ausgewählt. Weiterhin wurde der Algorithmus *Strukturlernen in (diskreten oder kontinuierlichen) DBN* für die dynamischen Bayesschen Netzwerke behandelt.
Den Abschluß der theoretischen Überlegungen bildete eine Einführung in die Integration von Vorwissen als eine wichtige Strategie bei der Rekonstruktion von genetischen Netzwerken.

Alle betrachteten Reverse Engineering Algorithmen wurden implementiert und zunächst ausführlich an Simulationsdaten getestet. Natürlich lassen sich die dabei im einzelnen erzielten Ergebnisse nicht direkt auf die Arbeit mit realen Expressionsdaten übertragen, denn wesentliche modellbedingte Limitationen der Algorithmen – beispielsweise die abstrakte Betrachtung der Genregulationsprozesse – kamen bei der Arbeit mit Simulationsdaten nicht zum Tragen. Es konnten jedoch wichtige Einblicke in das grundlegende Verhalten der Algorithmen in Abhängigkeit von verschiedenen Eigenschaften des zu rekonstruierenden Netzwerks und der verfügbaren Daten gewonnen werden:
Eine zunehmende Konnektivität des zugrundeliegenden Netzwerks erschwert die Identifizierung regulatorischer Einflüsse – erfährt eine Netzwerkkomponente nur einen einzigen regulatorischen Einfluß, so kann dieser besser nachgewiesen werden, als wenn neben ihm noch weitere regulatorische Einflüsse auf die Netzwerkkomponente einwirken. Bei steigender Netzwerkgröße liefern die Algorithmen neben den korrekt identifizierten regulatorischen Einflüssen eine zunehmende Anzahl an falsch identifizierten regulatorischen Zusammenhängen, denn sie müssen eine größere Menge an prinzipiell möglichen regulatorischen Einflüssen betrachten, um die auf eine Netzwerkkomponente einwirkenden Einflüsse festzulegen. Die Wahrscheinlichkeit, dabei einen Fehler zu begehen, steigt.
Weitere Simulationsexperimente konnten zeigen, daß ein sehr großer Umfang an Expressionsdaten notwendig ist, um möglichst viele der tatsächlich existierenden Einflüsse korrekt zu identifizieren. Außerdem sollten die gegebenen Daten keine Zustandsübergänge mehrfach enthalten. Die Beobachtung verschiedener zeitlicher Prozesse liefert mehr Informationen über das dynamische Verhalten des zugrundeliegenden Systems als die Betrachtung einer einzelnen Zeitreihe und kann das Reverse Engineering ebenfalls unterstützen. Deshalb sollte man versuchen, durch

eine geeignete Variation von extra- und intrazellulären Einflußfaktoren oder durch transiente Manipulationen bestimmter Gene unterschiedliche dynamische Prozesse des Genregulationsnetzwerks zu erfassen und damit möglichst viele Informationen über das dynamische Verhalten des Systems zu generieren.
Eine Untersuchung des Einflusses fehlerbehafteter Daten ergab, daß der negative Einfluß von Meßfehlern im Gegensatz zu den Schwierigkeiten, die sich durch vereinfachte und falsche Modellannahmen ergeben, kaum Auswirkungen auf die Güte der Ergebnisse zeigt. Deshalb ist zu erwarten, daß der negative Einfluß von Meßfehlern bei der Arbeit mit realen Expressionsdaten nur bedingt zum Tragen kommt.
Weiterhin konnte ein Vergleich der Ergebnisse der einzelnen Algorithmen eine Überlegenheit des Algorithmus *Strukturlernen in kontinuierlichen DBN* gegenüber den auf dem nichtlinearen Additiven Regulationsmodell basierenden Algorithmen bestätigen. Sie resultiert aus der Tatsache, daß dieser Algorithmus im Gegensatz zu den anderen Algorithmen die Schätzung der Gewichtsmatrix W mit einer expliziten Suche nach der wahren Struktur verbindet und dadurch den Reverse Engineering Prozeß maßgeblich unterstützt. Aufgrund der wahrscheinlichkeitstheoretischen Natur des kontinuierlichen DBN gelingt es ihm außerdem besser, mit Fehlern und Inkonsistenzen in den Daten umgehen. Im allgemeinen waren die Ergebnisse des *evolutionären Algorithmus* etwas besser als die des *BPTT* Algorithmus; der Algorithmus *REM* bei der Arbeit mit einem nichtlinearen Additiven Regulationsmodell konnte sich nicht gegen die anderen Algorithmen dieser Gruppe durchsetzen. Da die Generierung der Simulationsdaten auf der Basis des nichtlinearen Additiven Regulationsmodells stattfand, waren diese Algorithmen den übrigen Algorithmen bei der Arbeit mit Simulationsdaten überlegen, denn wichtige ihrer modellbedingten Limitationen kamen nicht zum Tragen.
Für die Algorithmen *Reveal* und *Strukturlernen in diskreten DBN* erschwerte beispielsweise die vereinfachende Annahme diskreter Expressionsraten das Identifizieren von regulatorischen Einflüssen zusätzlich. Dem Algorithmus *Reveal* gelang es vor allem bei steigendem Datenumfang, eine höhere Sensitivität zu liefern als der Algorithmus *Strukturlernen in diskreten DBN*. Dafür konnte dieser Algorithmus oftmals einen sehr hohen positiv prädiktiven Wert vorweisen. Eine fälschlicherweise angenommene Linearität regulatorischer Einflüsse beeinträchtigte den Reverse Engineering Prozeß für den Algorithmus *REM* bei der Arbeit mit einem linearen Additiven Regulationsmodell. Vor allem aufgrund einer in der Regel verhältnismäßig großen Anzahl an falsch identifizierten regulatorischen Einflüssen konnte dieser Algorithmus nicht gegen die anderen beiden Algorithmen überzeugen.
Ferner ergab die Analyse der Ergebnisse, daß durch die Kombination der Einzelergebnisse zweier Algorithmen vor allem die Anzahl der falsch identifizierten Einflüsse zum Teil erheblich gesenkt werden kann.
In einem Experiment zur Integration von Vorwissen wurde abschließend deutlich, daß sinnvoll eingesetztes Vorwissen die Ergebnisse der Algorithmen zum Teil erheblich verbessert.

Schließlich wurden alle Reverse Engineering Methoden auch an realen Expressionsdaten getestet. Das Anwendungsbeispiel vermittelte abschließend einen Eindruck, inwieweit die auf abstrakten Netzwerkmodellen basierenden Reverse Engineering Methoden die Identifizierung von regulatorischen Einflüssen aus derzeit verfügbaren Expressionsdaten überhaupt ermöglichen:
Grundsätzlich konnten in diesem Anwendungsbeispiel nur recht unbefriedigende Ergebnisse erzielt werden. Neben den wenigen korrekt identifizierten Einflüssen lieferten die Algorithmen eine recht hohe Anzahl zusätzlicher Einflüsse.
Gründe hierfür liegen zum einen in den modell- und algorithmenbedingten Limitationen der Reverse Engineering Algorithmen. Wesentliche Einschränkungen ergaben sich vor allem aufgrund der abstrakten Betrachtung der Genregulationsprozesse.
Berücksichtigt werden mußte aber auch, daß die Algorithmen nur solche regulatorischen Einflüsse identifizieren können, über die die gegebenen Expressionsdaten auch Informationen liefern. Deshalb stellt die Begrenzung der verfügbaren Daten, sowohl bezüglich des Umfangs als auch in Bezug auf den Datentyp – es waren nur die jeweiligen mRNA-Konzentrationen der Gene gegeben –, den eigentlichen Hauptgrund für die unbefriedigenden Ergebnisse dar. Durch diese Begrenzung wird die vereinfachte Betrachtung der Genregulationsprozesse überhaupt erst notwendig und ist daher weniger eine modellbedingte Limitation der Reverse Engineering Methoden, sondern vielmehr eine Limitation der verfügbaren Daten!
Die Ergebnisse aus der Arbeit mit Simulationsdaten über den Vergleich der Algorithmen konnten bei der Verwendung realer Expressionsdaten bestätigt werden. Da die Überlegenheit der auf dem nichtlinearen Additiven Regulationsmodell bzw. auf dem kontinuierlichen DBN – welches sich an diesem Modell orientiert – basierenden Algorithmen bei der Verwendung realer Expressionsdaten entfällt, war es jetzt auch möglich, die Ergebnisse dieser Algorithmen mit den Ergebnissen der übrigen Algorithmen zu vergleichen. Insbesondere hat sich dabei gezeigt, daß es dem Algorithmus *Reveal* in dieser konkreten Anwendung sogar besser als den mit dem Additiven Regulationsmodell bzw. kontinuierlichen DBN arbeitenden Algorithmen gelang, die bekannten regulatorischen Einflüsse zu identifizieren.

7.2 Ausblick

Im abschließenden Anwendungsbeispiel wurde deutlich, daß die Limitationen der Ergebnisse, die mit den vorgestellten Reverse Engineering Methoden erzielt werden können, sich vor allem durch die Begrenzung der verfügbaren Expressionsdaten ergeben. Es ist jedoch zu hoffen, daß eine Verbesserung und Weiterentwicklung der Microarray- und Protein-Array-Technologien sowie eine Massenproduktion dieser Biochips bessere Ergebnisse der Reverse Engineering Methoden ermöglichen werden: Zum einen liefern die derzeit verfügbaren Expressionsdaten aufgrund eines sehr geringen Datenumfangs nur unzureichende Informationen über das dynami-

sche Verhalten des zugrundeliegenden Systems. Durch eine Massenproduktion von Microarray-Chips werden diese in Zukunft immer kostengünstiger. Dies wiederum wird dann die Durchführung einer wesentlich größeren Anzahl an Expressionsexperimenten ermöglichen und den verfügbaren Datenumfang erheblich steigern. Zum anderen erfordert derzeit die Einschränkung der verfügbaren Daten auf mRNA-Konzentrationen eine abstrakte Betrachtung der Genregulationsprozesse, woraus die in dieser Arbeit beschriebenen Probleme resultieren. Die Weiterentwicklung und Verbesserung der Protein-Array-Technologie zur genomweiten Messung von Protein-Konzentrationen läßt hoffen, daß neben den mRNA-Konzentrationen bald auch die entsprechenden Protein-Konzentrationen der Gene zur Verfügung stehen und in den Reverse Engineering Prozeß integriert werden können. Ein erster Schritt dabei könnte die Protein-Konzentrationen zumindest auf der Ebene ihres regulatorischen Einflusses auf die Transkription der Gene berücksichtigen. Alle in dieser Arbeit vorgestellten Netzwerkmodelle und Reverse Engineering Algorithmen lassen sich recht einfach entsprechend erweitern. In einem nächsten Schritt können die Protein-Konzentrationen auch explizit in ein genetisches Netzwerkmodell integriert, die Produktion sowie der Abbau der Proteine modelliert und so die bei der Genexpression und Genregulation ablaufenden Prozesse biologisch realistischer dargestellt werden. Auf entsprechende, bereits existierende Ansätze (siehe u.a. [8]) kann zurückgegriffen werden, wenn die benötigten Daten erst verfügbar sind.

Obwohl das Anwendungsbeispiel zeigen konnte, daß auch eine diskrete Betrachtung der Expressionsraten durchaus die Identifizierung von regulatorischen Einflüssen ermöglicht, so sollte doch langfristig die Konzentration auf biologisch realistischeren Netzwerkmodellen liegen, die mit kontinuierlichen Expressionsraten arbeiten. Hier bieten sich vor allem die kontinuierlichen Dynamischen Bayesschen Netzwerke an, denn aufgrund ihrer wahrscheinlichkeitstheoretischen Natur können sie auch Meßfehler und Inkonsistenzen in den Daten berücksichtigen. Außerdem existieren für dieses Netzwerkmodell effiziente Reverse Engineering Algorithmen, wie beispielsweise der in dieser Arbeit vorgestellte Algorithmus *Strukturlernen in kontinuierlichen DBN*. Es ist sinnvoll, diese Reverse Engineering Methode weiterzuentwickeln und gezielte Verbesserungen sowohl an dem Netzwerkmodell als auch an dem Reverse Engineering Algorithmus vorzunehmen. Beispielhaft seien hier kurz drei Ansätze beschrieben:
Eine Möglichkeit zur Verbesserung bietet die Integration von verborgenen Variablen (*engl.: hidden variables*) in das kontinuierliche DBN [34]. Damit lassen sich Probleme lösen, die entstehen, wenn ein zentrales Regulatorgen im Reverse Engineering Prozeß nicht betrachtet wird – etwa aufgrund einer Fehlentscheidung bei der Selektion wichtiger Gene, oder weil das Expressionsverhalten des Regulatorgens in den Expressionsexperimenten gar nicht bestimmt wurde. Da die von ihm ausgehenden regulatorischen Einflüsse dann nicht nachweisbar sind, erklärt der Reverse Engineering Algorithmus das durch dieses Regulatorgen induzierte ähnliche Expres-

sionsverhalten der von ihm regulierten Gene durch regulatorische Einflüsse zwischen ihnen. Es entsteht eine hohe Anzahl falsch identifizierter Einflüsse. Eine verborgene Variable kann dieses Regulatorgen modellieren und die Identifizierung falscher regulatorischer Einflüsse verhindern.
Bei der Modellierung der Dynamik arbeitet das Netzwerkmodell mit einer konstanten Zeitverzögerung. Die auf ein Gen g_i einwirkenden regulatorischen Einflüsse werden durch eine Abhängigkeit der Expressionsrate $x_i(t)$ des Gens g_i zum Zeitpunkt t von den Expressionsraten $x_{i_1}(t-1)$,$x_{i_2}(t-1)$,...,$x_{i_k}(t-1)$ der dieses Gen regulierenden Gene zum vorangegangenen Zeitpunkt $t-1$ modelliert. Das Netzwerkmodell betrachtet dazu die kontinuierlichen Zufallsvariablen $\mathbf{X}(t-1)$, welche die Expressionsraten der Gene vor einem Zustandsübergang beschreiben, und die Zufallsvariablen $\mathbf{X}(t)$, die die aktualisierten Expressionsraten der Gene repräsentieren. Wie lange es im einzelnen dauert, bis eine Veränderung des Expressionsverhaltens eines Gens g_j auf die Expression eines von ihm regulierten Gen g_i Auswirkungen zeigt, hängt von den verschiedenen Prozessen ab, die bei der Genexpression und Genregulation ablaufen und ist individuell für zwei Gene g_j und g_i festgelegt. Durch die Einführung von individuellen Zeitverzögerungen könnte die biologische Realität genauer modelliert werden. Die Expressionsrate $x_i(t)$ des Gens g_i zum Zeitpunkt t ist dann von den Expressionsraten $x_{i_1}(t-\tau_{i,i_1})$,$x_{i_2}(t-\tau_{i,i_2})$,...,$x_{i_k}(t-\tau_{i,i_k})$ abhängig. Das Netzwerkmodell muß dafür einfach um entsprechende Zufallsvariablen $\mathbf{X}(t-2)$, $\mathbf{X}(t-3)$,...,$\mathbf{X}(t-\Delta t_{max})$ erweitert werden, die es erlauben, die aktuellen Expressionsraten der Gene auch mit weiter zurückliegenden Expressionsraten in Verbindung zu setzen.
Weiterhin ist es sinnvoll, als Ergebnis des Algorithmus *Strukturlernen in kontinuierlichen DBN* nicht nur das Modell mit dem höchsten Score zu betrachten. Oftmals haben viele Modelle einen recht ähnlichen Score und können damit die gegebenen Daten gleich gut erklären. Es sollten deshalb mehrere Modelle $B_i = \langle G_i, \Theta_i \rangle$ als Ergebnis zugelassen werden. Aus diesen sind dann regulatorische Einflüsse e mit einer hohen Posterior-Wahrscheinlichkeit zu selektieren und so eine endgültige Lösung zu konstruieren. Die Posterior-Wahrscheinlichkeit eines regulatorischen Einflusses e ergibt sich dabei aus [38]:

$$P(e|D) = \sum_{G_i} P(G_i|D)e(G_i) \tag{7.1}$$

Hierbei nimmt $e(G_i)$ den Wert 1 an, falls das Modell $B_i = \langle G_i, \Theta_i \rangle$ diesen regulatorischen Einfluß e beschreibt. Andernfalls ist der Wert von $e(G_i)$ 0.

Anhang A

Äquivalenz zwischen Maximum Likelihood Schätzung und der Methode der kleinsten Quadrate

Die Likelihood Funktion $L(\Theta_G : D|G)$ beschreibt die Wahrscheinlichkeit der Daten D in einem Bayesschen Netzwerk $B = \langle G, \Theta \rangle$ in Abhängigkeit von der Parameterinstanz Θ_G. Für ein kontinuierliches DBN, in dem die bedingten Wahrscheinlichkeitsverteilungen der Zufallsvariablen durch entsprechende Normalverteilungen beschrieben werden, ergibt sich:

$$\begin{aligned} & L(\Theta_G : D|G) \\ = \; & P(D|\langle G, \Theta_G \rangle) \\ = \; & \prod_{i=1}^{N} \prod_{m=1}^{M} f_i(y_i^m | \langle G, \Theta_G \rangle) \\ = \; & \prod_{i=1}^{N} \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(y_i^m - \mathbf{w}_i \mathbf{u}^{m^T})^2}{2\sigma^2} \right) \end{aligned} \tag{A.1}$$

Häufig arbeitet man mit dem Logarithmus dieser Funktion, um die Produkte zu Summen zu vereinfachen:

$$\begin{aligned} & \log \prod_{i=1}^{N} \prod_{m=1}^{M} \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(y_i^m - \mathbf{w}_i \mathbf{u}^{m^T})^2}{2\sigma^2} \right) \\ = \; & \sum_{i=1}^{N} \left(\log \frac{1}{(2\pi)^{\frac{M}{2}} \sigma^M} - \sum_{m=1}^{M} \frac{(y_i^m - \mathbf{w}_i \mathbf{u}^{m^T})^2}{2\sigma^2} \right) \end{aligned} \tag{A.2}$$

Um den Schätzer $\hat{\mathbf{w}}_i$ des Gewichtsvektors $\mathbf{w}_i$ zu bestimmen, muß die Likelihood Funktion bezüglich des Gewichtsvektors $\mathbf{w}_i$ maximiert werden. Dazu wird sie nach $\mathbf{w}_i$ differenziert und gleich dem Wert 0 gesetzt:

$$\sum_{m=1}^{M} (y_i^m - \mathbf{w}_i \mathbf{u}^{m^T}) \mathbf{u}^{m^T} = 0$$

$$\sum_{m=1}^{M} \mathbf{u}^{m^T} y_i^m = \sum_{m=1}^{M} \mathbf{u}^{m^T} (\mathbf{w}_i \mathbf{u}^{m^T})$$

$$\sum_{m=1}^{M} \mathbf{u}^{m^T} y_i^m = \sum_{m=1}^{M} \mathbf{u}^{m^T} (\mathbf{u}^m \mathbf{w}_i^T)$$

$$\sum_{m=1}^{M} \mathbf{u}^{m^T} y_i^m = \sum_{m=1}^{M} (\mathbf{u}^{m^T} \mathbf{u}^m) \mathbf{w}_i^T$$

$$\sum_{m=1}^{M} (\mathbf{u}^{m^T} \mathbf{u}^m)^{-1} \cdot \sum_{m=1}^{M} \mathbf{u}^{m^T} y_i^m = \mathbf{w}_i^T$$

$$(U^T U)^{-1} U^T \mathbf{y}_i = \mathbf{w}_i^T \tag{A.3}$$

Dies entspricht der Schätzung von w_i nach der Methode der kleinsten Quadrate [46].

Anhang B

Notation

Symbol	Erklärung
α	Irrtumswahrscheinlichkeit 1. Art eines statistischen Tests
Adj_G	Adjazenzliste von Graph G
Acc_G	Erreichbarkeitsliste von Graph G
β	Irrtumswahrscheinlichkeit 2. Art eines statistischen Tests
β_i	Biasfaktor von Gen g_i
BIC	Engl.: Bayesian information criteria
$BPTT$	Algorithmus *Backpropagation through time*
cDNA	Engl.: complementary DNA; einzelner DNA-Strang als Kopie eines mRNA-Strangs
D	Expressionsdaten
D_i	Zerfallskonstante von Gen g_i
DBN	Dynamisches Bayessches Netzwerk
$dimG$	Dimension der Struktur G (Anzahl der Parameter)
DNA	Engl.: desoxyribonuclein acid
$error_i(t)$	Fehler, der der Expressionsrate x_i von Gen g_i zum Zeitpunkt t bei der Generierung von Simulationsdaten zugefügt wird
$error_{i,max}(t)$	Maximal erlaubter Fehler $error_i(t)$ (betragsmäßig)
f_i	Boolesche Funktion f_i
$falseNeg$	Anzahl nicht identifizierter regulatorischer Einflüsse
$falsePos$	Anzahl falsch identifizierter regulatorischer Einflüsse
g_i	Gen i
G	Struktur eines Netzwerks
G_{akt}	Struktur des Netzwerks im aktuellen Iterationsschritt
G_{azyk}	Struktur eines azyklischen Netzwerks
G_{nb}	Nachbarstruktur einer Netzwerkstruktur G
G_{min}	Minimale Graphenstruktur

G_{prior}	Wahrscheinlichste Struktur des Genregulationsnetzwerks in Bezug auf das Vorwissen
H_{0,X,x_i}	Hypothese: x_i ist unabhängig von X
H_{1,X,x_i}	Hypothese: x_i ist abhängig von X
$H(X)$	Entropie einer Zufallsvariablen X
$H(X,Y)$	Gemeinsame Entropie zweier Zufallsvariablen X und Y
$H(X\|Y)$	Bedingte Entropie zweier Zufallsvariablen X und Y
IKIT	Institut für klinische Immunologie und Transfusionsmedizin (Universität Leipzig)
IL-6	Zytokin Interleukin-6
k	Konnektivität (Anzahl der Elternelemente eines Gens)
k_{max}	Maximal erlaubte Anzahl an Elternelementen eines Gens
$LR(X,Y)$	Likelihood Ratio von X und Y
max_i	Maximale Expressionsrate von Gen g_i
M	Datenumfang
M_{k_i}	Anzahl der Beobachtungen mit $x_i = k_i$
M_{j_i}	Anzahl der Beobachtungen mit $X = j_i$
M_{k_i}	Anzahl der Beobachtungen mit $x_i = k_i$ und $X_i = j_i$
$MI(X,Y)$	Wechselseitige Information zweier Zufallsvariablen X und Y
mRNA	Engl.: messenger ribonuclein acid
N	Anzahl der Netzwerkkomponenten (Gene)
N_{i,j_i,k_i}	Anzahl der Beobachtungen mit $X_i = k_i$ und $Pa(X_i) = j_i$
N_{i,j_i}	$N_{i,j_i} = \sum_{k_i} N_{i,j_i,k_i}$
pa_i	Zustand der Elternelemente eines Gens g_i
$Pa(X_i)$	Menge der Elternelemente einer Zufallsvariablen X_i in einem (Dynamischen) Bayesschen Netzwerk
ppW	Positiv prädiktiver Wert
r_i	Regulatorischer Input von Gen g_i
$rate$	Fehlerrate zur Generierung fehlerbehafteter Simulationsdaten
REM	Algorithmus *Reverse Engineering in Matrizen*
RT-PCR	Reverse Transcriptase Polymerase Chain Reaction
S	Suchraum der Netzwerkstrukturen
$S(\cdot)$	Sigmoidalfunktion
SAGE	Serial Analysis of Gene Expression
$truePos$	Anzahl korrekt identifizierter regulatorischer Einflüsse
U	Inputmatrix
w_{ij}	Gewicht zur Spezifikation des regulatorischen Einflusses von Gen g_j auf Gen g_i
W	Gewichtsmatrix
x_i	Zustand von Gen g_i
$\mathbf{x}$	Systemzustand des Genregulationsnetzwerks
$\mathbf{x_i}$	Attraktorzustand des Genregulationsnetzwerks nach der

	Manipulation von Gen g_i
X_i	Zufallsvariable in einem Bayesschen Netzwerk; modelliert Zustand von Gen g_i
$X_i[t]$	Zufallsvariable in einem Dynamischen Bayesschen Netzwerk; modelliert Zustand von Gen g_i zum Zeitpunkt t
Y	Outputmatrix
η	Lernrate in einem neuronalen Netzwerk
Θ	Parametermenge eines (Dynamischen) Bayesschen Netzwerks
θ_{i,x_i,pa_i}	$\theta_{i,x_i,pa_i} = P(X_i = x_i \mid Pa(X_i) = pa_i)$; Parameter in einem (Dynamischen) Bayesschen Netzwerk

Literaturverzeichnis

[1] Abkowitz, J.L., Catlin, S.N. und Guttorp, P.: Evidence that hematopoiesis may be a stochastic process in vivo. *Nature Medicine 2:190-197, 1996.*

[2] Akutsu, T., Kuhara, S., Maruyama, O. und Miyano, S.: Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. *Proceedings of the ninth annual ACM-SIAM symposium on discrete algorithms:695-702, 1998.*

[3] Akutsu, T., Miyano, S. und Kuhara, S.: Identification of genetic networks from a small number of gene expression pattern under the Boolean network model. *Proceedings of the Pacific Symposium on Biocomputing, 1999.*

[4] Akutsu, T., Miyano, S. und Kuhara, S.: Algorithms for inferring qualitive models of biological networks. *Proceedings of the Pacific Symposium on Biocomputing, 2000.*

[5] Ando, S. und Iba, H.: Inference of gene regulatory model by genetic algorithms. *Proceedings of the 2001 IEEE Congress on Evolutionary Computation, 2001.*

[6] Arkin, A., Ross, J. und McAdams, H.H.: Stochastic kinetic analysis of developmental pathway birfurcation in phage λ-Infected *Escherichia coli* Cells. *Genetics 149:1633-1648, 1998.*

[7] Brockhaus - Die Enzyklopädie: in 24 Bänden - 20.,überarbeitete und aktualisierte Auflage, 1998.

[8] Chen, T., He, H.L. und Church, G.M.: Modeling gene expression with differential equations. *Proccedings of the Pacific Symposum on Biocomputing,vol.4:29-40, 1999.*

[9] Chickering, D.M.: Learning Bayesian networks is NP-complete. *Learning from Data: Artificial Intelligence and Statistics, Vol.5: 121-130, 1996.*

[10] Conant, R.C.: Extended dependency analysis of large systems part I: Dynamic analysis. *International Journal of General Systems, 14:97-123, 1988.*

[11] Dempster, A.P., Laird, N.M. und Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society 39:1-38, 1977.*

[12] D'haeseleer, P., Liang, S. und Somogyi, R.: Genetic network inference: from coexpression clustering to reverse engineering. *Bioinformatics,16:707-726, 2000.*

[13] D'haeseleer, P.: Reconstructing gene networks from large scale gene expression data. *Dissertation, The University of New Mexico, 2000.*
http://www.cs.unm.edu/∼patrik/networks/networks.html

[14] EMBL: SAGE for beginners. *http://www.embl-heidelberg.de/info/sage*

[15] Friedman, N.: Learning Bayesian networks in the presence of missing values and hidden variables.*Proeceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence and Statistics (UAI), 1997.*

[16] Friedman, N., Murphy, K. und Russell, S.: Learning the structure of dynamic probabilistic networks. *Proeceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence and Statistics (UAI): 139-147. Morgan Kaufmann Publishers, San Francisco, CA, 1998.*

[17] Friedman, N.: The Bayesian Structural EM Algorithm. *Proccedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence and Statistics (UAI), 1998.*

[18] Friedman, N., Linial, M., Nachman, I. und Pe'er, D.: Using Bayesian networks to analyze expression data. *Journal of Computational Biology 7:601-620, 2000.*

[19] Friedman, N. und Nachman,I .: Gaussian process networks. *Proc. of the Sixteenth International Conference on Uncertainty in Artificial Interlligence (UAI), 2000.*

[20] Friedman, N. und Goldszmidt, M. and Lee,T .J.: Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting. *Proc. of the 15th International Conference on Machine Learning, 1998.*

[21] GlaxoSmithKline: Introduction to Genetics
http:/genetics.gsk.com/overview.htm, 2003

[22] Gross, J. und Yellen, J.: Graph theory and its applications. *CRC Press Boca Raton, London, New York, Washington,D.C, Seite 372, 1999.*

[23] Heckerman, D. und Geiger, D.: Learning Gaussian networks. *Technical Report MSR-TR-94-10, Microsoft Research, 1994.*

[24] Heckerman, D. und Geiger, D.: Learning Bayesian networks: a unification for discrete and Gaussian domains. *Proc. of the 11th Conference on Uncertainty in Artificial Interlligence (UAI), 1995.*

[25] Heckerman, D.: A tutorial on learning Bayesian networks. *Technical Report MSR-TR-95-06, Microsoft Research, 1995.*

[26] Heidrich, K., Kretschmar, A., Pfeier, G., Henze, c., Löffler, D., Koczan, D., Thiesen, H., Burger, R., Gramatzki, M. und Horn, F.: Close correlation between Stat3 target gene expression and IL-6-dependent survival of multiple myeloma cells.

[27] Hertz, J.: Statistical issues in reverse engineering of genetic networks. *Poster fpr Pacific Symposium on Biocomputing, http://www.nordita.dk/∼hertz/papers/dgshort.ps.gz, 1998.*

[28] Hofmann, R. und Tresp, V.: Discovering structure in continuous variables using Bayesian networks. *Advances in Neural Information Processimg Systems 8, 1996.*

[29] Ideker, T.E., Thorsson, V. und Karp, R.M.: Discovery of regulatory interactions through perturbation: inference and experimental design. *Proceedings of the Pacific Symposium on Biocomputing 5:302-313, 2000.*

[30] Jacobs, R.A.: Increased rates of convergence through learning rate adaption. *Neural Networks 1,4:295-307, 1998.*

[31] Jong, H.D.: Modeling und Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology 9:67-103, 2002.*

[32] Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretic Biology 22:437-467, 1969*

[33] Kauffman, S.A.: The Origins of Order: Self organization and selection in evolution. *Oxford University Press, Oxford, 1993.*

[34] Koller, D.: Lecture notes and readings. *http://robotics.stanford.edu/∼koller*

[35] Lexikon der Biologie. *Spektrum Akademischer Verlag GmbH Heidelberg· Berlin, 2001.*

[36] Liang, S., Fuhrman, S. und Somogyi, R.: Reveal, a general reverse engineering algorithm for inference of genetic network architeckture. *Proceedings of the Pacific Symposium on Biocomputing, vol.3:18-29, 1998.*

[37] Lottspeich, F.: Bioanalytik. *Hrsg.: Zorbas, H., Spektrum Akademischer Verlag GmbH Heidelberg· Berlin, Kapitel 36: Gezielte Genmodifikation(921-939)& Kapitel 38: Überexpression(959-977), 1998.*

[38] Mathäus, D.: Analyzing gene-expression data with Bayesian networks. *Diplomarbeit, Technische Universität Graz, Institut für Elektro- und Biomedizinische Technik, 2002.*

[39] McAdams, H.H. und Arkin, A.: Stochastic mechanisms in gene expression. *Proceedings of the National Acadamy of Sience, USA 94:814-819, 1997.*

[40] Meyer-Lüerßen, D.: Einsatz von Biochips eröffnet neue Dimensionen in diagnostischer Praxis. *Hrsg.: VDGH Verband der Diagnostica-Industrie e.V., Diagnostik im Gespräch, 2003.*

[41] Miller, G.: Note on the bias of information estimates. *Information theroy in psychology, ed.: Quastler,H., 1955.*

[42] Missal, K.: Modellierung von Reverse Engineering Strategien zur Identifizierung genetischer Netzwerke aus unvollständigen Genexpressionsdaten. *Diplomarbeit, Universität Leipzig, Institut für Informatik, 2003.*

[43] Mitchell, M.: An introduction to genetic algorithms. *pages:65-79, The MIT Press, 1999.*

[44] Mjolsness, E. Sharp, D.H. und Reinitz, J.: A connectionist model of development. *Journal of Theoretic Biology 152:429-454, 1991.*

[45] Mülhardt, C.: Der Experimentator: Molekularbiologie/Genomics. *Spektrum Akademischer Verlag GmbH Heidelberg· Berlin, 3.Auflage, 2002.*

[46] Murphy, K. und Mian, S.: Modeling gene expression data using Dynamic Bayesian Networks. *Technical Report, University of California, 1999.*

[47] Onami, S., Kyoda, K.M., Morohashi, M. und Kitano, H.: The DBRF method of inferring a gene network from large-scale steady state gene expression data. *Foundations of System Biology, ed.:Kitano,H.:59-75, 2001.*

[48] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. *Morgan Kaufmann, San Francisco, CA, USA, 1988.*

[49] Preisler, H.D. und Kauffman, S.: A proposal regarding the mechanism which underlies lineage choice during hematopoiectic differentiation. *Leukemia Research 23:685-694, 1999.*

[50] Projektbeschreibung des Teilprojekts A13: Funktionelle Analyse von Interleukin-6-Zielgenen und Interleukin-6-abhängige Expressionsmuster in Lymphozyten. *http://www.uni-leipzig.de/∼izkf/inhalte/a13.htm*

[51] Römpp Lexikon Biotechnologie. *Hrsg.: Prof. Dr. Dellweq, H., Prof. Dr. Schmid, R.D. und Prof. Dr. Trommer, W.E., Georg Thieme Verlag Stuttgart·New York, 1992.*

[52] Sachs, L.: Angewandte Statistik. *Springer Verlag Berlin·Heidelberg·New York, 7.Auflage, 1992.*

[53] Schöneburg, E., Heinzmann, F. und Feddersen, S.: Genetische Algorithmen und Evolutionsstrategien. *Addison-Wesley, 1994.*

[54] Schwarz, G.: Estimating the dimension of a model. *The Annals of Statistics 6:461-464, 1978.*

[55] Shannon, C.E.: A mathematical theory of communication. *Bell System Technical Journal 27:379-423 and 623-656, 1948.*

[56] Somogyi, R. und Fuhrman, S.: Distributivity, a general information theoretic network measure, or why the whole is more than the sum of its parts. *Proc. International Workshop on Information Processing in Cells and Tissues, 1997.*

[57] Szallasi, Z.: Genetic network analysis - from the bench to computers and back. 2^{nd} *International Conference on Systems Biology, 2001.*

[58] The R project for statistical computing. *http://www.r-project.org*

[59] Tietze, F.: Bio-Datenbanken. *Seminar Biodatenbanken, Prof. Dr. Rahm,E., Universität Leipzig, WS2002/03.*

[60] van Rooij, A.J.F., Jain, L.C. und Johnson, R.P.: Neural network training using genetic algorithms. *World Scientific Publishing Co. Pte. Ltd., 1996.*

[61] van Someren, E.P., Wessels, L.F.A and Reinders, M.J.T.: Genetic network models: a comparative study. *Proceedings of SPIE; Micro-array:Optical Technologies and Informatics (BIOS01), 2001.*

[62] Velculescu, V.E., Zhang, L., Vogelstein, B. and Kinzler, K.W.: Serial analysis of gene expression. *Sciences 270, 5235: 484-487, 1995.*

[63] Wagner, A.: How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps. *Bioinformatics 17:1183-1197, 2001.*

[64] Weaver, D.C., Workman, C.T. und Stormo, G.D.: Modeling regulatory networks with weight matrices. *Proceedings of the Pacific Symposium on Biocomputing 4:112-123, 1999.*

[65] Xu, L. und Jordan, M.I.: On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computatations 8:129-151, 1996.*

Erklärung

Ich versichere, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, den 9. November 2004 Antje Müller